Self-Collision Detection and Prevention for Humanoid Robots

James Kuffner[1,2] Koichi Nishiwaki[1] Satoshi Kagami[2]
Yasuo Kuniyoshi[1] Masayuki Inaba[1] Hirochika Inoue[1]

1 Dept. of Mechano-Informatics, School of Information Science and Technology, Univ. of Tokyo.
7–3–1, Hongo, Bunkyo-ku, Tokyo, 113–8656, Japan.
{kuffner,nishi,kuniyosh,inaba,inoue}@jsk.t.u-tokyo.ac.jp

2 Digital Human Lab., National Institute of Advanced Industrial Science and Technology
2-41-6, Aomi, Koto-ku, Tokyo, 135-0064, Japan.
s.kagami@aist.go.jp

Abstract

We present an efficient approach to self-collision detection suitable for complex articulated robots such as humanoids. Preventing self-collisions is vital for the safe operation of robots that generate body trajectories online. Our approach uses a fast distance determination method for convex polyhedra in order to conservatively guarantee that a given trajectory is free of self-collision. Experimental results using an online joystick control application for the humanoid robot "H7" demonstrate the feasibility and effectiveness of the method.

1 Introduction

In order for humanoid robots to become practical, they must be able to operate safely and reliably. *Self-collisions* occur when one or more of the links of a robot collide. Self-collisions can result in damage to the robot itself, or through a loss of balance or control, cause human injury or damage to its surrounding environment. Thus, detecting and avoiding self-collisions is fundamental to the development of robots which can be safely operated in human environments.

This paper describes an efficient geometric approach to detecting link interference suitable for complex articulated robots such as humanoids. We rely on fast, feature-based minimum distance determination methods for convex polyhedra[9] in order to conservatively guarantee that a given trajectory is free of self-collision. Threshold values can be set on the allowable minimum distance between links in order to provide a safety margin that accounts for errors in modeling and control. Full body trajectories can be checked in advance for potentially self-colliding postures prior to being executed on the robot.

Our previous experiments concerned detecting and preventing leg collisions[7]. In our current implementation, the minimum distances between all possible relevant body link pairs (Figure 1(a)) for a 30 DOF humanoid (435 pairs) can be calculated in approximately 2.5 milliseconds on average on an 866 MHz Dual Pentium III (see Section 5). Although we have focused on detecting self-collisions for biped humanoids, the technique can generally be applied to any robot with articulated appendages (arms or legs). It is also applicable to detecting inter-robot collisions for multiple manipulators which share a common workspace (e.g. crowded factory workcells).

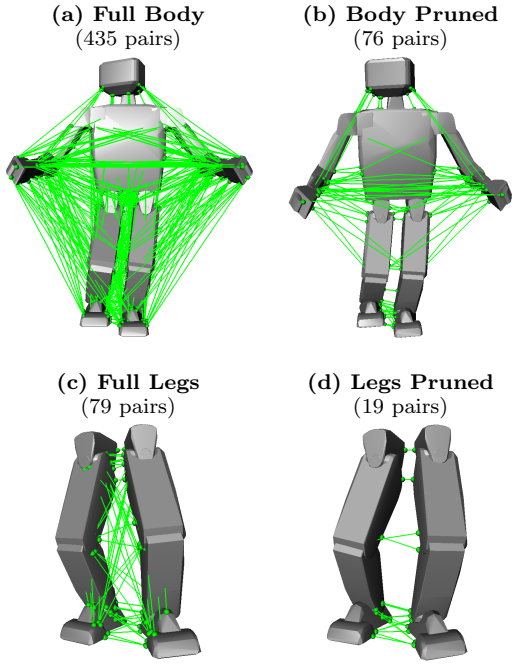


Figure 1: Four different self-collision detection modes for the H7 humanoid. Minimum distances are maintained between active pairs of link *protective hulls*.

2 Background

Collision detection in robotics has arisen primarily in the context of obstacle avoidance and path planning, in which the robot geometry is tested for collision with geometric models of environment obstacles. Articulated robots must also avoid self-collision, though most serial-chain manipulators are designed mechanically to minimize potential link interference. Self-collision is typically not an issue for mobile robots.

In the case of serial-chain manipulators, immediately adjacent links cannot collide if proper joint angle limits are defined. Other pairs of links can be ruled out due to geometric reachability constraints. The remaining link pairs must be checked for collision explicitly. For a manipulator with six or seven degrees of freedom (DOF), performing these remaining checks can usually be accomplished using any number of efficient model-based collision detection algorithms from computational geometry (e.g. [8, 11, 3, 1, 9, 4, 2]).

Humanoid robots, however, pose a particular challenge for self-collision detection. A humanoid generally consists of a tree of connected links. This tree can be conceptually viewed as a set of five serial chain manipulators (2 arms, 2 legs, and 1 neck-head chain) all attached to a free-floating trunk. Each chain must avoid self-collision as well as collisions with the trunk and all other chains. Assuming that joint limits prevent collision between a given link and its parent link, the number of pairs P that must be checked explicitly in tree-like structure with N links is given by:

$$\begin{aligned} P &= (\sum_{i=1}^{N-1} i) - (N-1) = \sum_{i=1}^{N-2} i \\ P &= \frac{N^2 - 3N + 2}{2} \end{aligned} \quad (1)$$

The H7 humanoid developed in our laboratory has a total of 31 links and 30 DOF. For $N = 31$, the maximum number of remaining pairs to be checked according to Equation 1 is $P = 435$. The burden of this computation is no longer trivial.

Clearly, the number of pairs which must be checked in practice can be substantially reduced by considering kinematic reachability constraints. For example, it is impossible for the neck and head links of H7 to collide with the leg links. Through heuristic or exhaustive search methods, a table of all potentially colliding pairs of links can be pre-computed [6]. However, even after eliminating unnecessary pairs, the number of remaining active pairs is still considerable. In the case of H7, a total of 76 pairs must be checked in order to verify an arbitrary posture is free of self-collision (see Figure 1). For verifying walking trajectories for H7 that involve only the leg joints, checking fully all links yields 79 pairs, while checking only selected links still demands 19 pairs. Clearly, efficient underlying techniques for collision detection between pairs of geometric primitives are required.

3 Interference Detection

Collision detection can be included under the more general term *interference detection*. Interference detection and the related problem of minimum distance determination between two or more geometric objects has been the subject of extensive research in both robotics and computer graphics. The important features of a given algorithm include: 1) efficiency, 2) generality (e.g. ability to handle non-convex models, models with holes, polygon "soups", etc.) 3) numerical stability and robustness, and 4) exploitation of spatial and temporal coherence. Performance tradeoffs in terms of both speed and memory exist between each of these features.

Trajectory Sampling: In its simplest form, interference detection returns a binary result (whether or not two or more geometric objects overlap). This implies that checking for collision between objects following continuous motion trajectory necessitates either: 1) computing the *swept volumes* of the object motions and checking for interference, or 2) discretizing the trajectory into a finite set of samples which are individually tested for collision.

Since swept volume computations are difficult and expensive, discretization is usually preferred due to simplicity. However, *regardless of the discretization resolution one selects*, it is always possible to construct a case in which a potentially dangerous collision goes undetected due to an insufficient number of samples.

Bounds and Collision-free Guarantees: The advantage of knowing a conservative measure of the minimum distance between two objects over a purely binary collision detection result, is that it allows one to formulate *collision-free guarantees* over bounded relative motions of the objects.

Let $dist(\mathcal{A}, \mathcal{B}, T_{\mathcal{AB}}) \mapsto \Re$ be a function that returns a real number representing a lower bound on the minimum distance between two objects $\mathcal{A}$ and $\mathcal{B}$ at a given relative transformation $T_{\mathcal{AB}} \in SE(3)$. Let $\tau(t) \mapsto T_{\mathcal{AB}}$ be a continuous function that parameterizes the relative motion between $\mathcal{A}$ and $\mathcal{B}$ in $\Re^3$.

For our humanoid robot, bounds on the maximum joint velocities $|\dot{q}| < \dot{q}_{max}$ determine bounds on the maximal velocities of the robot geometry in $\Re^3$ via the Jacobian $\partial x = J(q)\partial q$ which maps velocities in the joint space into the workspace. Thus, given a continuous joint trajectory $\tau(t) \mapsto q$ and an interval of time Δt, the maximal displacement $x_{max}(q, \dot{q}, \dot{q}_{max}) \mapsto \Re$ in the workspace can be computed along with the minimum distance information to guarantee collision-free

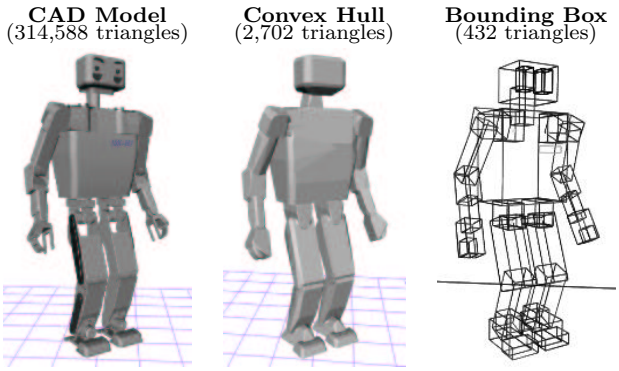


Figure 2: Approximate geometric models of H7.

Figure 3: Protective Convex Hull for the head link.

motion during the time interval Δt. For efficiency, conservative bounds can also be formulated in advance by using a fixed D_{max} per Δt representing the maximal displacement for all postures, though this is currently not done in our system.

Protective Hulls: Computational efficiency and conservative bounds are also considered when representing the link geometry for interference detection. Approximate convex *protective hulls* of each link are derived from the original CAD models, which represent inherently closed surface models of solid objects. These protective hulls serve as conservative approximations to the complicated geometry of individual links (see Figure 3). The hulls completely enclose the underlying geometry, and provide a *safety margin* around each link.

Our experiments have shown these hulls to nicely approximate the geometry relative to other possible bounding volumes (such as bounding boxes). Should a link have severe non-convex geometrical features, it can always be subdivided into a rigid collection of convex pieces. This was not necessary for our robot. Figure 2 shows three different approximate geometric models of the H7 humanoid robot (30 DOF, 137cm, 55kg). The left image shows tesselated CAD data for each link, the center image shows the link convex protective hulls (used for our experiments), and the right image shows the link bounding boxes.

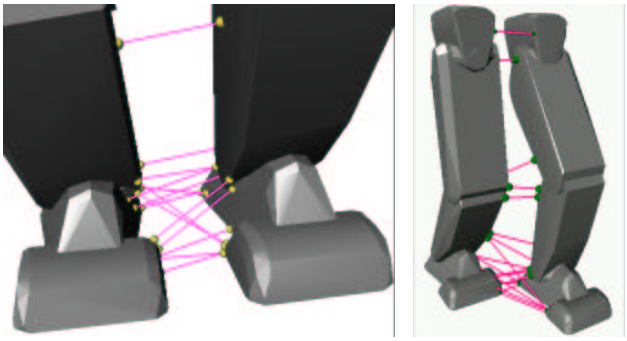


Figure 4: Maintaining pairs of closest points between leg links.

Minimum Distance Determination: The convex nature of the protective hulls allows us to utilize fast minimum distance determination methods for convex polyhedra. We selected the Voronoi-clip (*V-clip*) algorithm due to its robustness and availability[9]. V-clip is a feature-based method which improves upon the Lin-Canny algorithm[8], and has been shown to compare favorably to simplex-based algorithms such as Enhanced-GJK[1]. For convex polyhedra, V-clip does not need to construct hierarchies of bounding volumes like other methods [11, 3]. The running time does not depend on the distance between objects; only on their geometric complexity and motion relative to the previous query. Since our motion trajectories are continuous, V-clip is able to exploit spatial and temporal coherency in order to update minimum distance values and pairs of closest feature points in "almost constant" time (see [9] for details).

We track the closest points between each active pair of links over the course of an entire trajectory, and verify that a positive minimum distance is maintained at all times (see Figure 4). Forward kinematics is used to calculate the global position and rotation of each link from the leg joint angles. The user can set threshold values for the minimum distances considering errors in modeling and control, relative to the safety margin provided by the protective hulls.

4 Safe Walking

Most existing walking biped robots employ local feedback from force/torque sensors and gyroscopes in order to maintain dynamic balance. Using this sensor data and an approximate dynamic model, the joints of the robot can be adjusted online to satisfy constraints, such as center of gravity or ZMP constraints. (e.g.[13, 5, 12]). This feedback can then be used to successfully follow pre-calculated walking patterns, as well as walking trajectories generated online[10].

Although these methods are able to maintain dynamic stability, they often *do not guarantee safe mo-*

tion. While trying to satisfy balance constraints, they may inadvertently cause one leg of the robot to collide with the other leg ("leg interference"). As mentioned previously, such collisions can potentially cause the robot to fall, causing human injury, damage to itself, or damage to surrounding objects.

Collision Constraints: Theoretically, kinematic constraints aimed at avoiding self-collision could be incorporated directly into the balancing scheme. However, such an approach is currently impractical due to the numerous complex constraints induced by the geometry and kinematics of the robot links.

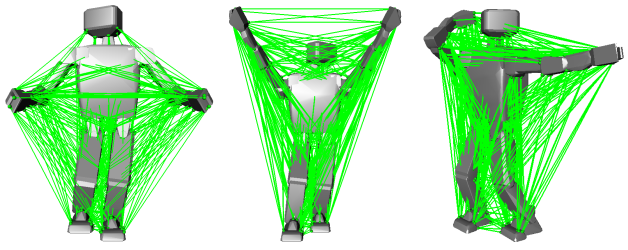
Detection and Prevention: Rather than attempting to satisfy self-collision constraints directly, we have focused on detecting potentially dangerous trajectories *before* they are executed by the robot. If detected, the robot can attempt to generate a new trajectory, or proceed with a safe stopping behavior. Using this approach, we aim to build internally-consistent walking systems that are safe, reliable, and practical. Our current implementation (described in the following section) uses an online joystick control application that combines dynamically-stable walking pattern generation with self-collision checking to produce safe walking trajectories.

5 Experiments

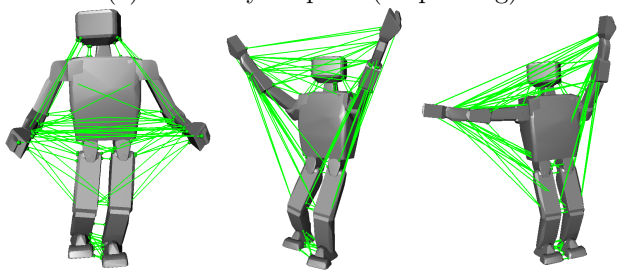
We have implemented a prototype self-collision detection module in C++ on an 866 MHz Dual Pentium III PC running RT-Linux. Given an input posture and a robot model (kinematics, joint hierarchy, protective hull geometry), the minimum distance between all sets of *active links* is calculated. The set of active pairs is determined by the current *self-collision checking mode*. All modes are shown in Figure 1. Example postures for each mode are shown: Full body (Figure 5), and Legs (Figure 6).

Computation Time: Based on the current mode, we verify that a single posture or an entire trajectory is free of potentially dangerous self-collisions. For a single posture of the robot (7 joint angles for each leg), 19 closest-feature pairs can be updated in less than 0.13 msec on average, including the forward kinematics calculations. This calculation was performed 10,000 times using varying cyclic leg configurations (both colliding and disjoint) of typical walking trajectories. A summary of the computation times for repeated runs of for each mode is shown in Table 1.

User Interface: A graphical interface is used to view the results of the minimum distance computations (see Figure 7). Lines connecting the closest

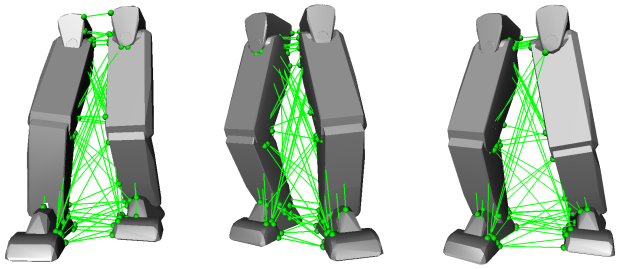


(a) Full body all pairs (no pruning)

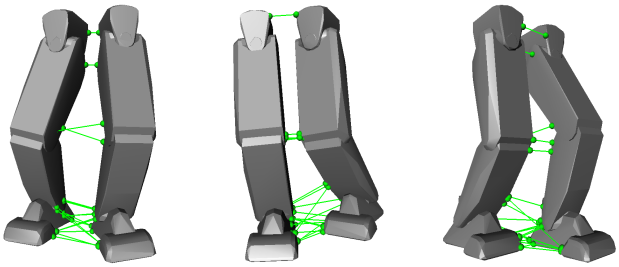


(b) Full body pruned

Figure 5: Full body self-collision detection.



(a) Legs all pairs (no pruning)



(b) Legs pruned

Figure 6: Legs self-collision detection.

Checking Mode	links	pairs	time (msec)
Full Body	31	435	2.442
Body Pruned	31	76	0.429
Full Legs	14	79	0.441
Legs Pruned	14	19	0.128

Table 1: Computation time for single-posture check including forward kinematics($N = 10,000$ trials).

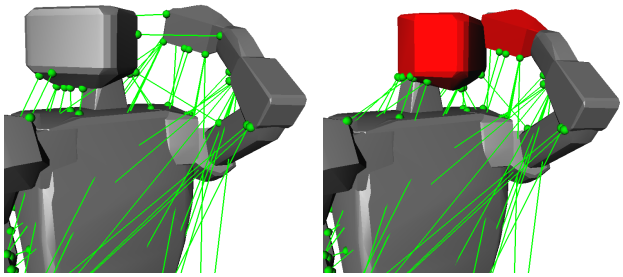


Figure 7: Detail of minimum distance calculation (Pruned body mode): non-colliding (left), colliding (right).

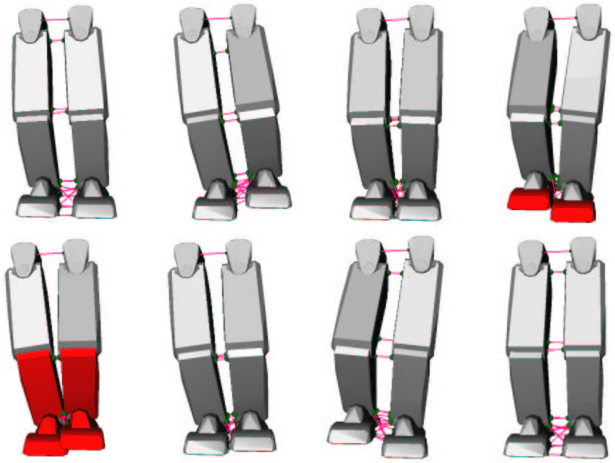


Figure 8: In-place step with in an inter-leg collision.

points between pairs of links are drawn. If the minimum distance between two links falls below zero (collision), the pair is shown in red. Figure 8 and Figure 9 illustrate two trajectories which are dynamically-stable, but result in inter-leg collisions.

Online Joystick Control: We have implemented an online joystick control application running under RT-Linux that combines dynamically-stable walking pattern generation with self-collision checking to produce safe walking trajectories. An overview of the relevant system components in shown in Figure 10.

The robot begins with a safe 3-step stopping trajectory (a trajectory which will bring the robot to a complete stop using three steps). Such a trajectory we refer to as an *Emergency Stopping Trajectory*, or EST. The timing for each step trajectory is approximately one second. The robot begins by executing the first step of the current 3-step EST. During that time, the joystick is queried and a new dynamically-stable EST to bring the robot from the end of the first step is calculated. The new motion is calculated according to the commanded joystick direction, and will bring the robot to a full stop (3 steps total). This new trajec-

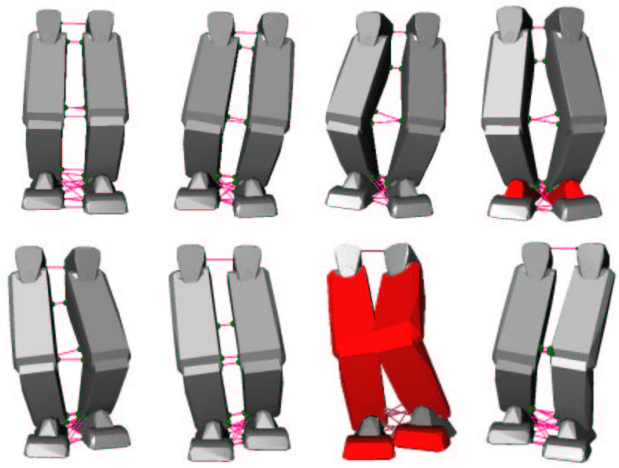


Figure 9: Forward-turning step with inter-leg collisions.

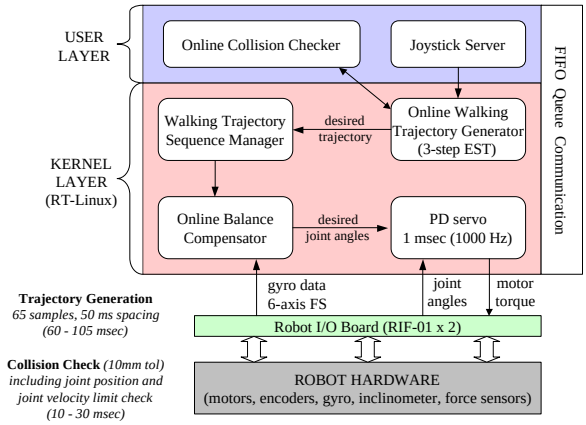


Figure 10: Software and Control System Components.

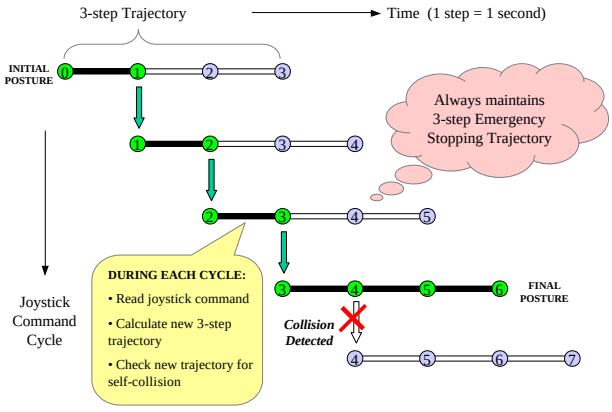


Figure 11: 3-step EST trajectory generation timeline.

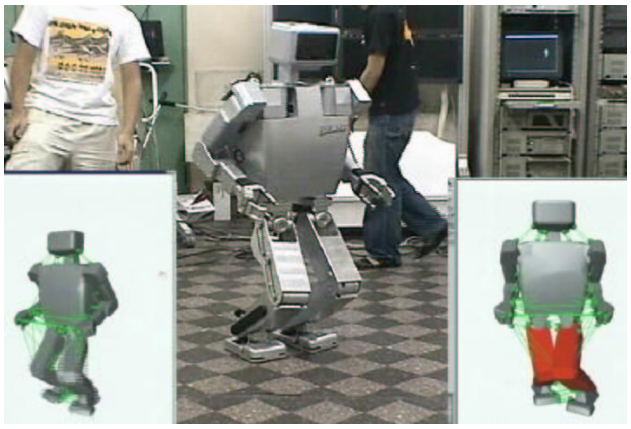


Figure 12: Online self-collision checking during joystick control experiment.

tory is sent via FIFO queue to a self-collision checking server which calculates the overall minimum distance between all active pairs. If the minimum distance falls below the given threshold, the new trajectory is discarded and the robot continues to follow the previous EST and comes to a halt. A graphical diagram of the behavior of the walking system is shown in Figure 11.

Figure 12 shows an online joystick experiment. The self-collision checking module (bottom left and right windows) detects a potentially dangerous knee-knee collision, and automatically causes the robot to follow a safe stopping trajectory. Experimental timing results collected from the RT kernel over hundreds of walking steps reveal that self-collision checking (Legs Pruned mode) for a three-step trajectory can be performed in roughly 10-30 msec on average using our current CPU hardware.

6 Discussion

We have presented an overview of an efficient approach to detecting self-collision for humanoid robots aimed at providing safety guarantees for full-body trajectories (both offline and online). We employ efficient minimum distance determination methods for maintaining the closest pair of features on conservative convex protective hull models of the leg links. We have incorporated the algorithm into an online joystick control application for the H7 humanoid robot, and demonstrated its efficiency and effectiveness.

For future work, we would like to use the software to automatically calculate active pairs for given joint angle ranges in order to reduce the combinations of pairs that must be checked dynamically. We are also currently investigating the use of alternative minimum distance determination methods[2], which may allow us to use non-convex protective hulls, or even the CAD data directly for situations in which the *exact* minimum distance information is required.

Acknowledgments

This research is supported in part by a Japan Society for the Promotion of Science (JSPS) Postdoctoral Fellowship for Foreign Scholars in Science and Engineering.

References

[1] S. Cameron. Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'97)*, pages 3112–3117, April 1997.

[2] S. A. Ehmann and M. C. Lin. Accelerated proximity queries between convex polyhedra by multi-level voronoi marching. In *Proc. of 2000 IEEE/RSJ Int. conf. on Intelligent Robots and Systems (IROS'00)*, 2000.

[3] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *SIGGRAPH '96 Proc.*, 1996.

[4] L. Guibas, D. Hsu, and L. Zhang. H-Walk: Hierarchical distance computation for moving convex bodies. In *Proc. ACM Symposium on Computational Geometry,*, pages 265–273, 1999.

[5] Kazuo Hirai. Current and future perspective of honda humanoid robot. In *Proc. of 1997 IEEE/RSJ Int. conf. on Intelligent Robots and Systems (IROS'97)*, pages 500–508, 1997.

[6] F. Kanehiro and H. Hirukawa. Online self-collision checking for humanoids. In *Proc. 19th Annual Conf. of Robotics Society of Japan*, Tokyo, Japan, September 2001.

[7] J.J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Efficient leg interference detection for biped robots. In *Proc. 19th Annual Conf. of Robotics Society of Japan (RSJ'01)*, Tokyo, Japan, September 2001.

[8] M. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, U.C. Berkeley, Dept. of Electrical Eng. and Comp. Sci., Berkeley, CA, 1993.

[9] B. Mirtich. VClip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, July 1998.

[10] K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online generation of desired walking motion on humanoid based on a fast method of motion pattern that follows desired zmp. In *19th Annual Conf. of Robotics Society of Japan*, 2001.

[11] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3324–3329, 1994.

[12] Y. Tamiya, M. Inaba, and H. Inoue. Realtime balance compensation for dynamic motion of full-body humanoid standing on one leg. *Journal of the Robotics Society of Japan*, 17(2):268–274, 1999.

[13] J. Yamaguchi, A. Takanishi, and I. Kato. Development of a biped walking robot compensating for three-axis moment by trunk motion. In *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS '93)*, pages 561–566, 1993.