An Efficient Method for Collision Detection and Distance Queries in a Robotic Bridge Maintenance System

J. Xu[1], D.K. Liu[1], and G. Fang[2]

[1] ARC Centre of Excellence for Autonomous Systems (CAS), Faculty of Engineering, University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia
{jiex, dkliu}@eng.uts.edu.au
[2] School of Engineering, University of Western Sydney, Locked Bag 1797, Penrith South DC 1797, Australia
g.fang@uws.edu.au

Abstract. When applying autonomous industrial robotic systems in an unknown/partially known or cluttered environment, mapping and representation of the environment as well as collision detection becomes crucial. Existing techniques in these areas are generally complex and computationally expensive to implement. In this paper an efficient sphere representation method is introduced for environment representation, collision detection and distance queries. In particular, this method is designed for the application in an autonomous bridge maintenance system. Simulation results show that this method is effective in environment representation and collision detection. Furthermore, the proposed method is also computationally efficient for real-time implementation.

1 Introduction

Industrial robots have been widely used in manufacturing, painting, welding and material handling. Most applications require industrial robots to perform repetitive jobs. Once they are programmed, they will repeat the same movements by following pre-planned paths. When there are uncertainties in either the robot work environment or the robot locations, the robot needs to "know" or "resolve" these uncertainties before it can take any actions.

Therefore, when industrial robots are to be used in the maintenance of large and complex infrastructures, such as steel bridges, the traditional systems need to be extended to deal with the major challenges that include:

1) Time critical: the robot has to be able to detect potential collisions and take actions in real-time.
2) Varying environment and non repetitive tasks: since the robot will be facing different structures or different aspects of the same structure, the robot is required to move in different paths. Furthermore, due to the non repetitive nature of the tasks, the planning and collision detection for the robot have to be performed in real-time.

T.-J. Tarn et al. (Eds.): Robot. Weld., Intellige. & Automation, LNCIS 362, pp. 71–82, 2007.
springerlink.com

3) Complex and tight environment: the structure, or the environment may be complicated or cluttered, and the work environment is small and crowded compared to the robotic work place in a workshop. Thus, efficient collision detection and avoidance become critical and difficult problems that must be solved in real-time.
4) Partially known environment: even though the CAD drawings of a structure or bridge always exist, it is difficult to guarantee that all refurbishments and upgrades are recorded in the drawings. Thus, the environment is most likely to be partially known.

These challenges have led to the need of developing effective and efficient methods to accurately map the environment, detect potential collisions and plan collision-free paths.

In general, the path planning in such applications as infrastructure maintenance uses the geometry information of the structure to generate an effective and efficient trajectory to cover the structure surface that needs to be treated. Any potential collision must be detected to verify if the entire trajectory is "valid", i.e., collision-free.

When using the robot for infrastructure maintenance, such as steel bridge cleaning, the accuracy of the robotic system is not as critical as in traditional applications such as welding and material handling. This gives some flexibility in developing the control algorithm.

In this research, an efficient sphere based approach is introduced for environment representation, virtual surface generation, collision detection, and robot arm pose calculation for applications in an autonomous robotic bridge maintenance system, based on the environmental map obtained from mapping. In this approach, the environment and the robotic arm are represented with a number of spheres, which give a simple and quick way for collision detection and distance queries. This approach will be presented in Section 3 after the review of related work in Section 2. Section 4 presents the simulation results, which is followed by discussions and conclusions in Section 5.

2 Related Works

Collision detection and distance measurement is a time-consuming but important task when an industrial robot is applied in a crowded environment. The computational time of intersection test and distance measurement highly depends on the types of primitives, the number of primitives (or the number of objects) and the algorithms used [1].

The general approach to solve this problem is to use hierarchy of bounding volume which is suitable for performing 3D proximity queries between general polyhedra [2]. The volumes can be discrete orientation polytope (k-DOPs) [3], oriented bounding boxes (OBB) [4], spheres [5]-[10], axis aligned bounding boxes (AABB) [4][11], swept sphere volumes (SSV) [12][13], convex hull [4][14], ellipsoids [15] or spherical shells [16], depending upon the requirements of computational time and accuracy.

Although OBBs, k-DOPs and ellipsoids can describe an object more precisely than spheres and AABBs in most cases, they are more time consuming [4]. Hubbard [5][6] used Octree and medial-axis surface to generate spheres and Bradshaw [7] improved it by using adaptive medial-axis approximation. Pobil et al. [8]-[10] used spherical approximation to represent a robot arm and heuristic methods to build the hierarchy and refine sub-region when it is necessary.

Distance query is more computationally expensive than collision detection because of additional geometric analysis and distance calculation [18]. Johnson et al. [19] used OBB and AABB to measure the distance under a lower-upper bound tree framework. One benefit of OBB is that the tree does not need to be recomputed when the orientation of an object changes [17]. However, the algorithm for building an OBB Tree is more expensive than AABB in terms of computational time.

Okada et al. [11] and Caselli et al. [20] evaluated most of the proximity query packages (I-COLLIDE, RAPID, V-COLLIDE, SOLID, V-Clip, PQP, and SWIFT) using the C/C++ platform, and found the performance was highly dependent on the problem to be solved.

Reichenbach et al. [21] used OBB to perform on-line trajectory re-planning. Redon et al. [22][23] introduced a general method, continuous collision detection (CCD), to detect possible collisions between two collision-free way-points.

In general, the sphere based approach is computationally more efficient than the rectangle based methods. As a result, in the applications where accuracy is less critical than efficiency, a sphere representation should be used.

3 The Efficient Spherical Approach

As discussed in Section 2, the spherical approximation approach has been investigated by many researchers for object and robotic arm representation. In this section, a simplified spherical approach is presented for environment representation, distance queries and robot joint pose calculation in an application to steel bridge maintenance.

3.1 The Efficient Spherical Representation

Conventionally, spheres have been used to approximate an object by minimising the difference between the real object and the virtual object represented by the spheres. In this research, however, the difference between the virtual and the real objects is used as a buffer to prevent the collision between the environment and the robotic system. A set of spheres can create a safety strip between the robot and the environment. Thus, instead of trying to minimise the difference, we determine the difference value based on the robot speed and required safety distance between the robot arm and the environment.

Fig. 1 shows the spherical representation of a robot arm and a environment consisting of I-beams and bridge deck in a bridge structure. Five spheres with different radius are used to represent the robotic arm. A number of spheres with the same radius are used to represent I-beams and the bridge deck.

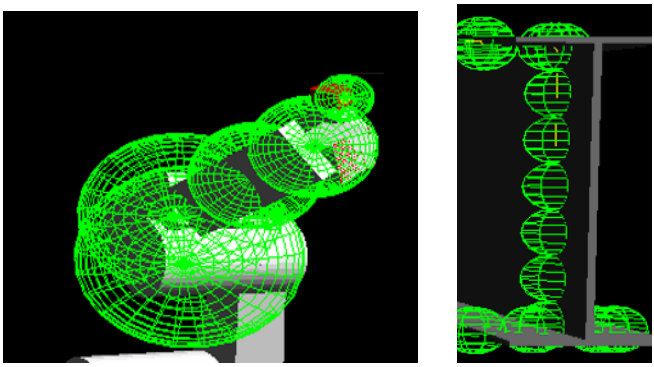

Fig. 1. Spherical representation of a robot arm (left) and I-beams and bridge deck (right)

With this spherical representation of the environment, the buffer (the shaded area in Fig. 2a) is used for collision avoidance. The minimum difference in distance, D_s, is defined as the safety distance (Fig. 2a) which is greater than or equal to the preset requirement D_{smin}, i.e. $D_s \geq D_{smin}$. With this safety distance requirement, the distance between the two adjacent spheres, d, can be calculated as:

$$d = 2 \times \sqrt{(R_1 + R_2)^2 - (R_2 + D_s)^2} \tag{1}$$

Where R_1 *is* the radius of the sphere representing the environment and R_2 is the smallest radius among the 5 spheres representing the robot arm (Fig. 2b). The centres of the environment spheres, e.g., O_1 and O_2, are located on the real surface.

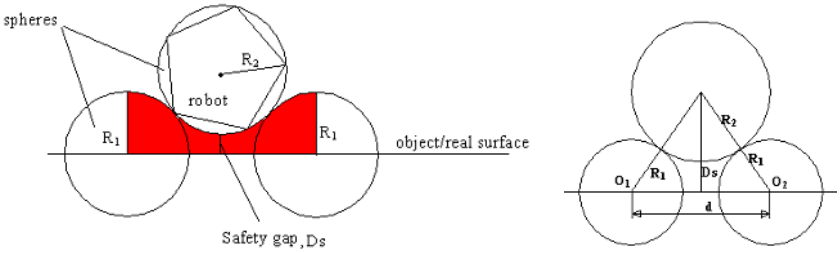

Fig. 2. (a) The buffer area used as the safety area (left) and (b) distance calculation (right)

In real-time collision detection and distance queries between the robot arm and the environment presented by the spheres, the distance can be simply calculated according to the centres of the robot spheres and the environmental spheres:

$$D = \min\left\{\sqrt{(X_{ri} - X_{ej})^2 + (Y_{ri} - Y_{ej})^2 + (Z_{ri} - Z_{ej})^2}\right\} \tag{2}$$
$$\forall i = 1,2,\ldots,N_r\,;\, j = 1,2,\ldots,N_e$$

where (X_{ri}, Y_{ri}, Z_{ri}) is the centre of the robot arm spheres, N_r is the number of spheres used to represent the robot arm. (X_{ej}, Y_{ej}, Z_{ej}) is the centre of a sphere representing the environment and close to the robot arm. N_e is the number of spheres in the environment and close to the robot arm. Although this is an approximation, safety distance D_{smin} is guaranteed.

The number of spheres used to represent the environment has a significant effect on the computational efficiency of collision detection and distance queries. The number and the radius are determined based on the complexity of the environment, safety distance requirement, maximum moving speed of the robot arm, etc. On a flat surface and with slow robot moving speed, the number of spheres, N_s, for the surface representation can be approximately calculated as:

$$N_s = \left(\frac{L_1}{d} + 1 \right) \times \left(\frac{L_2}{d} + 1 \right) \tag{3}$$

Where, L_1 and L_2 are the length and width of the surface, respectively.

3.2 Virtual Surface

With the spherical representation of an environment, surface of the buffer area forms what can be called a *virtual surface* (Fig. 3a) which prevents the robot arm from penetrating into the buffer area. Fig. 3b shows a part of a virtual surface in 3D. The virtual surface is a hard constraint to the robot arm movement in a 3D environment and is helpful for analysing any possible collisions and the motion of the robot arm.

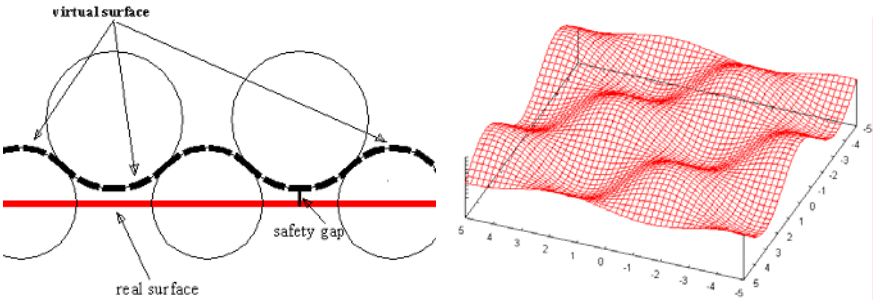

Fig. 3. (a) Virtual surface generation (left) and (b) a virtual surface in 3D (right)

3.3 Optimal Pose Calculation Based on the Spherical Representation

The distance between the robot arm and the environment, D, is used to determine the joint movement. Instead of using the traditional inverse kinematics method, an optimisation based pose calculation approach is applied. This optimisation method can effectively avoid singularity which occurs commonly in the inverse kinematic solutions. It can also obtain the optimal pose for joint movements by satisfying various constraints. The objective function of the pose selection is:

$$\min_{\theta_j} \quad (f_0 + f_1 + f_2 + f_3 + f_4 + f_5) \tag{4}$$

Where, θ_j are the joint angles of the robot (j = 1, 2, …, 6) and subjected to the constraints of the robot joint movements. $f_0, f_1, \ldots, f_5$ are the measurement vectors (or residual vectors).

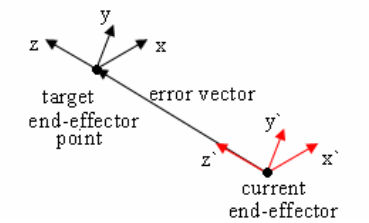

Fig. 4. x, y and z represent target orientation; x`, y` and z` represent current orientation

The difference between the end-effector/nozzle's current position and the target position is defined as *error vector* (Fig.4). End-effector/nozzle's position and orientation are expressed in a 4×4 matrix:

$$T = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where the top left 3×3 elements define the orientation in x, y and z directions, p_x, p_y and p_z represent the position.

$$f_0 = \text{(error vector)} \bullet \text{(x` direction)} \tag{6}$$

$$f_1 = \text{(error vector)} \bullet \text{(y` direction)} \tag{7}$$

f_0 and f_1 are used to make sure that the nozzle is pointing at the target. In addition, the angle between z' direction and z direction is constrained in the range of [-45°, +45°]. This constraint is expressed in the following equation:

$$f_2 = e^{-k(\alpha+45^o)} + e^{k(\alpha-45^o)} \tag{8}$$

where α is the angle between z` and z, k is a non-negative slope coefficient. f_3 represents the distance between the nozzle and its target position:

$$f_3 = \|\text{distance vector}\| \tag{9}$$

f_4 represents the constraints of robot joint angles and is expressed in the following equation:

$$f_4 = \sum_{j=1}^{6} (e^{-k_j(\theta_j - \min_j)} + e^{k_j(\theta_j - \max_j)}) \tag{10}$$

where θ_j is the jth joint angle while min_j and max_j represent the minimum and the maximum allowed joint angles of the jth joint of the robot. f_5 is a function of the distance between the robot arm and the environment represented by the spheres:

$$f_5 = \sum_{i=1}^{n} e^{-k \times D_i} \tag{11}$$

where, n is the number of distance queries; and D_i is the distance between the robot arm and the environment and is calculated in Equation (2).

By minimising the objective functions, the angular displacement of each joint, θ_j, of the robot arm at any instance can be obtained. A least square algorithm [24] is employed to solve the optimal pose selection problem.

4 Simulation Results

Simulations are conducted based on a Denso robot (model VM-6083D-W). It is assumed that the robot is used to clean a steel bridge that consists of I-beams and a deck (Fig.5a). The Denso robot has 6 joints and is placed between the two I-beams under the deck. The robot joint ranges of θ_1 to θ_6 are listed in Table 1.

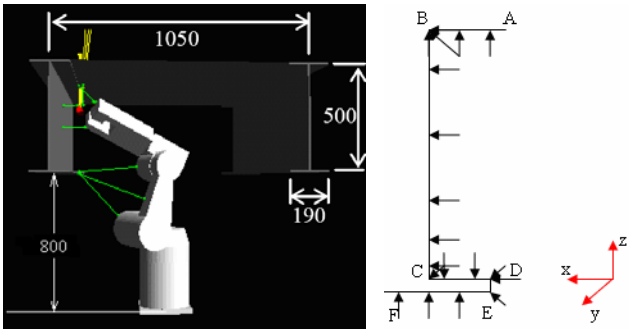

Fig. 5. (a) The robot and the bridge structure (left); (b) a path from A, B, C, D, E to F along the I-beam, arrows point to the surface represent the orientation of the nozzle (right)

At a position, the robot will clean one section of the bridge. The paths of the robot arm for cleaning the section are designed by a path planner. Fig.5b shows an example path along the surface of an I-beam from point A, B, C, D, E, to F. The arrows in the figure represent the orientation (perpendicular to the surface) of the cleaning nozzle.

Table 1. Denso VM-6083D-W joint angle range (degree)

	Minimum	Maximum
θ_1	-170	170
θ_2	-90	135
θ_3	-10	165
θ_4	-185	185
θ_5	-120	120
θ_6	-360	360

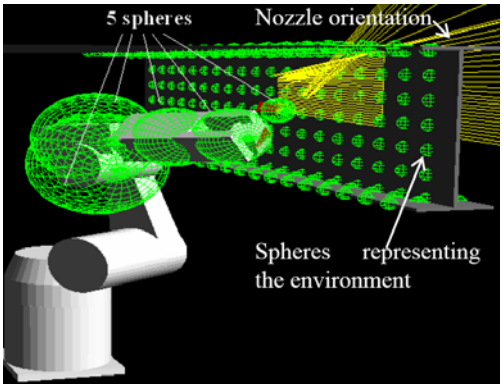

Fig. 6. A path and nozzle orientation in a strictly constrained structural environment

88 pre-defined waypoints are set along the path from A to F. The robot moves the cleaning nozzle installed on the robot end-effector along the path. In this movement, the nozzle orientation can vary between 45° to 90° in order to maintain the cleaning quality and productivity. The challenge of this operation is that the robot arm needs to move from the top side to the bottom side of the I-beam with a 180°orientation change in a tightly constrained environment. Fig.6 shows the robot arm represented by 5 spheres and the spherical approximation of the environment. A path from the top is also shown with nozzle orientation displayed.

The radii of the 5 spheres of the robot arm are 0.2m, 0.18m, 0.10m, 0.044m, and 0.11m, respectively, based on the size of the robot arm. The spheres presenting the I-beams and the deck have the radius of 0.02m. When the minimum safety distance is D_{smin}=0.006m and R_2 is 0.044m (the 5th radius), d is 0.08m according to Equation (1). Thus, d should be less than or equal to 0.08m.

When the robot moves along the path, the distances between the robot arm and the environment are calculated for collision detection at the waypoints. In addition, if any one of the 6 joint angles changes significantly (e.g., more than 10°) between the consecutive waypoints, collision should also be checked during the robot movement between these two waypoints although there is no collision at the two waypoints. This collision detection is simply done by adding more points between the two and then checking those added points.

Fig.7 shows how the robot tracks a path and satisfies the nozzle orientation requirement without any collision. The four figures in Fig.7 show different robot arm positions and nozzle orientations. The robot joint movements at each time step are calculated by the optimal pose calculation method discussed in Section 3.3.

The simulation results show that the movements of the 6 joints from waypoint to waypoint are continuous and smooth. The spherical approach based optimal pose selection method can find appropriate poses for joints to make the robot motion smooth and collision free. Even when the robot is dealing with the bottom surface of the I-beam

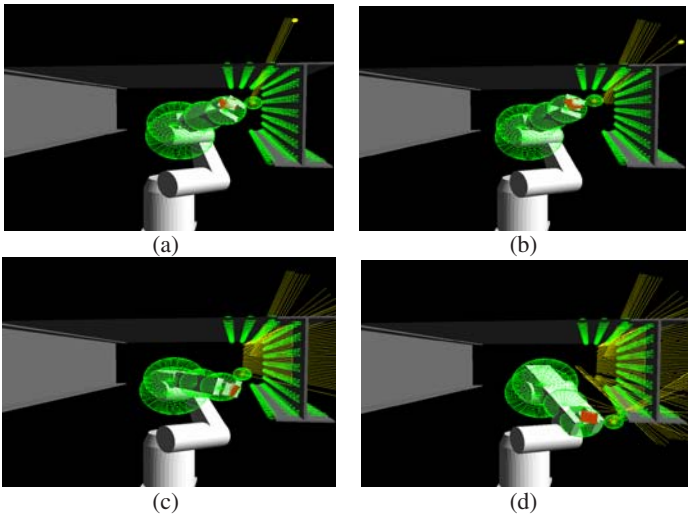

Fig. 7. The robot moves from the top to the bottom of the I-beam

where the nozzle orientation changes sharply, the selected robot movement still does not cause any collisions. Fig. 8a shows how the joint angle changes along the path from waypoints 1 to 88. It can be seen that the 6th joint has significant angular changes in its movement from waypoints 14 to 20 and from waypoints 75 to 77. Collision detection in these two parts of the path confirms that it is collision free.

The distance between the robot arm and the environment during the movement of the robot along the path is shown in Fig.8b. Seven distance queries are recorded during the process. They are d_1 to d_5 representing the distance between the five spheres of the robot arm and the environment, d_6, the distance from the end-effector of the robot arm to the environment and d_7 the distance from the nozzle (installed on the end-effector) to the environment. It can be seen that d_6 and d_7 increase sharply from waypoints 74 to 88 while others decrease. This is because the nozzle and the end-effector of the robot arm move away from the deck when they approach the bottom surface of the I-beam. It can also be seen that all the distances are greater than 0 at any time along the path, which demonstrates that there is no collision.

In order to investigate the efficiency of the spherical approach introduced in this research, PQP approach is also applied. Both approaches were used for distance queries and collision detections on different paths. Due to the space limitation of the paper, only the results of six (6) randomly selected paths are shown. The computational time required by the two approaches is presented in Table 2. It can be seen that, for the application discussed in this research, the represented spherical approach is more efficient than the PQP. It is clear from the table that the average reduction of computational time is about 80%.

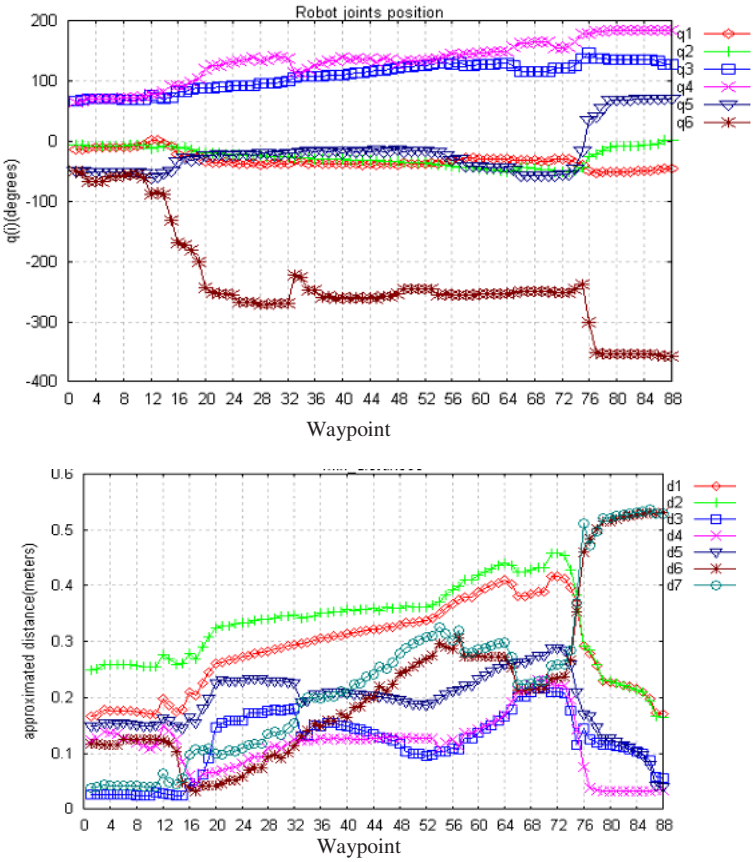

Fig. 8. (a) The changes of angles (unit: degree) of joints q1 to q6 (upper), (b) distances between the spheres of the robot arm and the environment (bottom)

Table 2. Efficiency comparison

Computational time (sec) / Path / Methods	1	2	3	4	5	6
PQP	89.2	83.8	82.4	81.0	83.3	75.4
The approach in this paper	18.9	16.6	14.5	13.5	9.6	13.8
Reduction (%)	78.7	80.2	82.5	83.3	88.4	81.7

5 Conclusions

In this paper, a simplified spherical approach is studied for application in a robotic bridge maintenance system. This approach is shown to be computationally efficient for distance queries and collision detection in complex and constrained environments. Although this efficiency gain is at the cost of a reduced accuracy in representing the

robot and the environment, it is acceptable for applications where the computational time is more critical than accuracy. This simplified spherical approach can be used to represent the robot and the environment with much less number of spheres. It can also generate a virtual surface that can be used to evaluate the safety distance requirement in collision avoidance. When combined with an optimization method, this sperical representation approach can be successfully used to calculate the joint movements of a robot arm without collision with the environment.

We are working now to improve the method to: (1) achieve an automatic sphere generation strategy to minimize human intervention; (2) test in more complex environments; and (3) develop collision avoidance strategies using the distance information obtained from this approach.

References

1. Schwarzer, F., Saha, M., and Latombe, J. C., Adaptive Dynamic Collision Checking for Single and Multiple Articulated Robots in Complex Environments, *IEEE Transactions on Robotics*, Vol. 21, No. 3, June 2005.
2. Weghorst, H., Hopper, G., and Greenberg, D. P., Improved Computational Methods for Ray Tracing, *ACM Transaction on Graphics*, Vol. 3, No. 1, January 1984.
3. Klosowski, J.T., Held, M., Mitchell, J.S.B., Sowizral, H., and Zikan K., Efficient Collison Detection Using Bounding Volume Hierarchies of k-DOPs, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 1, 1998.
4. Gottschalk, S., Lin, M.C., and Manocha, D., OBBTree: A Hierarchical Structure for Rapid Interference Detection, *Proc. of 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ACM press, 1996.
5. Hubbard, P.M., Interactive Collision Detection, *Proc. of IEEE Symposium on Research Frontier in VR*, 1993.
6. Hubbard, P.M., Approximating Polyhedra with Spheres for Time-Critical Collision Detection, *ACM Transaction on Graphics*, Vol. 15, No. 3, 1996.
7. Bradshaw, G., O'Sullivan, C., Adaptive medial-axis approximation for sphere-tree construction, *ACM Transactions on Graphics,* Vol. 23, 2004.
8. Del Pobil, A.P., Serna, M. A., Llovert, J., A new representation for Collision Avoidance and Detection, *Proceeding of the IEEE International Conference on Robotics and Automation,* 1992.
9. Del Pobil, A.P., Pkrez, M., Martinez, B., A Practical Approach to Collision Detection between General Objects, *Proceeding of the IEEE International Conference on Robotics and Automation,* 1996.
10. Martinez-Salvador, B., Perez-Francisco, M., Del Pobil, A.P., Collision Detection between Robot Arms and People, *Journal of Intelligent and Robotic Systems,* Kluwer, Vol. 38, 2003
11. Okada, K., Inaba, M., Inoue, H., Real-Time and Precise Self Collision Detection System for Humanoid Robots, *Proceeding of the IEEE International Conference on Robotics and Automation*, 2005.
12. Larsen, E., Gottschalk, S., Lin, M.C., Manocha, D., Fast Distance Queries with Rectangular Swept Sphere Volumes, *Proceeding of the IEEE International Conference on Robotics and Automation*, 2000.
13. Larsen, E., Gottschalk, S., Lin, M.C., Manocha, D., *Fast Distance Queries with Swept Sphere Volumes*, Department of Computer Science, UNC, Rep. TR99-081, 1999.

14. Ehmann, S., Lin, M.C., Accelerated Distance Computation between Convex Polyhedra by Multi-Level Marching, Technical Report, Department of Computer Science, UNC at Chapel Hill, 1999.
15. Kimoto, T., Yasuda, Y., Shape Description and Representation by Ellipsoids, *Signal Processing: Image Communication*, Vol. 9, No. 3, Elsevier Science, March 1997.
16. Füfzig, C., Ullrich, T., Fellner, D.W., Hierar chical Spherical Distance Fields for Collision Detection, *Computer Graphics and Applications*, IEEE, 26(1), 2006.
17. Sanchez-Ante, G., *Single-Query Bi-Directional Motion Planning with Lazy Collision Checking*, PhD thesis, Department of Computer Science, Instituto Tecnologico Y De Estudios, Mexico.
18. Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic, 1991.
19. Johnson, D.E., Cohen, E., A Framework for Efficient Minimum Distance Computations, *Proceeding of the IEEE International Conference on Robotics and Automation*, 1998.
20. Caselli, S., Reggiani, M., Mazzoli, M., Exploiting Advanced Collision Detection Libraries in a Probabilistic Motion Planner, *Journal of WSCG*, 10(1-3), 2002.
21. Reichenbach, T., Kovacic, Z., Collision-Free Path Planning in Robot Cells Using Virtual 3D Collision Sensor, *Cutting Edge Robotics*, ISBN 3-86611-038-3, pp. 683~704, ARS/plV, 2005.
22. Redon, S., Kim, Y.J., Lin, M.C., Manocha, D., Fast Continuous Collison Detection for Articulation Models, *ACM Symposium on Solid Modeling and Application*, 2004.
23. Redon, S., Fast Continuous Collision Detection and Handling for Desktop Virtual Prototyping, *Virtual Reality*, Springer, Vol. 8, No. 1, 2004.
24. Nocedal, J. and Wright, S. J., Numerical Optimization, Springer-Verlag, New York, 1999