Automatic Generation of Sphere Hierarchies from CAD Data

Joe Pitt-Francis*
Computing Laboratory
Oxford University, Parks Road
Oxford OX1 3QD, UK
Joseph.Pitt-Francis@comlab.ox.ac.uk

Roy Featherstone†
Dept. of Engineering Science
Oxford University, Parks Road
Oxford OX1 3PJ, UK
roy@robots.ox.ac.uk

Abstract

A sphere hierarchy is a data structure that approximates the shape of a given object using a collection of spheres. They can be used for collision detection, motion planning and other related applications. This paper presents an algorithm for constructing an efficient sphere hierarchy from a geometric model of an object, as supplied by a CAD system or geometric modeller; and it presents a branching factor analysis based on a general measure of the efficiency of a sphere hierarchy.

1 Introduction

In this paper, we present an algorithm for creating sphere hierarchies automatically from geometric data stored in a CAD system or geometric modeller. The main features of this algorithm are:

1. it copes with general geometry;
2. the resulting sphere hierarchies are efficient; and
3. the interface to the CAD system (or solid modeller) is kept simple.

The efficiency of a sphere hierarchy is the average computational efficiency of an intersection-testing algorithm given that sphere hierarchy as an argument. We define a measure of efficiency and use it to perform a branching-factor analysis of the hierarchy, in order to determine the optimal branching factor.

Sphere hierarchies are a well-known technique for approximating the shape of a given body using a collection of spheres. The hierarchical organization imparts two useful properties: it allows efficient intersection tests and minimum distance calculations between hierarchies (or between sphere hierarchies and other shape representations); and it allows a trade-off between accuracy and computational effort. These properties make sphere hierarchies attractive for use in collision-detection and path-planning applications. A selection of works on sphere hierarchies can be found in [1, 3, 4, 6, 10, 11, 13, 14, 16]. Related work on hierarchies of other shapes can be found in [2, 5, 8], and on non-hierarchical sphere-based representations in [9, 12, 15].

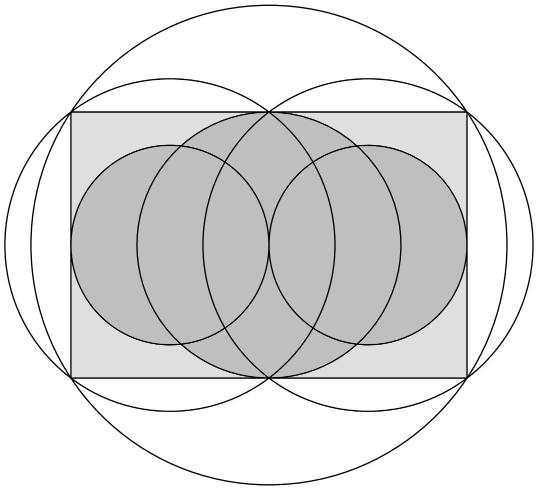

Figure 1: A sphere hierarchy built with concentric pairs of inner and outer spheres.

We are using sphere hierarchies in connection with a new technique called 'swept bubbles' [6, 7, 16] that combines shape descriptions of moving bodies with kinematic models of their motions, in order to perform efficient intersection tests in the Cartesian product of Euclidean and configuration spaces; but the sphere hierarchies produced by our algorithm can obviously be used in any of the usual applications.

Sphere hierarchies come in a variety of different forms. The type that we discuss in this paper con-

*Supported by EPSRC Research Grant number GR/L32408.
†Supported by EPSRC Advanced Fellowship number B92/AF/1466.

sists of concentric pairs of inner and outer spheres, organized into a binary tree (see Figure 1). The choice of binary trees in preference to trees with higher branching factors is motivated by the results of the branching-factor analysis, which concludes that a branching factor of two is generally the most efficient. The advantage of using paired inner and outer spheres is that they provide both an interior representation and a simple measure at each node in the hierarchy of how accurately it approximates its portion of the body's shape. The latter is needed by the swept-bubble algorithms. Applications are free to ignore the inner spheres if they wish.

The algorithm described in this paper generates sphere hierarchies by the method of recursive binary subdivision of the original shape, with optimizations, and is essentially an improved version of the algorithms described in [14, 16].

In the rest of this paper, we present first the subdivision algorithm for generating sphere hierarchies, then the efficiency measure and the branching factor analysis, and finally some results.

2 The Subdivision Algorithm

The sphere hierarchy of an individual body (*e.g.* a robot link) is generated by the method of recursive binary subdivision, which proceeds as follows. First, the smallest enclosing sphere is found; then the largest concentric inner sphere that lies inside the body. If the difference between the two radii is less than a specified value then the representation is deemed to be sufficiently accurate, and the algorithm returns. Otherwise, it cuts the body into two pieces and constructs a hierarchy around each; and the root nodes of these hierarchies become the children of the current node.

Enclosing spheres are calculated by requesting from the modeller the set of vertices of the body to enclose (which is assumed to be a polyhedron), and calculating the smallest sphere that encloses this set of points.

Inner spheres are more complicated, because there are two cases to consider: if the centre of the sphere lies inside the body then the radius is the distance from the centre to the nearest point on the original body's boundary; but if the centre lies outside the body then we define a *negative-radius sphere* whose radius is the negative of the minimum distance from the centre to the body. (The usual semantics of sphere intersection can be extended in a consistent and useful manner to include negative-radius spheres.) Notice that we use the *original* body for this calculation, not the recursively chopped up piece.

The calculation proceeds as follows. First, the modeller is asked for the minimum distance between the centre and the original body. If this value is positive then the inner radius is set to the negative of the minimum distance; but if it is zero then we calculate a positive inner radius as the minimum perpendicular distance from the centre to the plane equation of any face of the original body. This calculation produces the correct inner radius if the original body is convex, otherwise it produces an underestimate. (We hope to replace this algorithm with a better one soon.) To save on communication traffic between the hierarchy builder and the solid modeller, the entire set of plane equations is requested once at the start of the construction process, and stored for later use.

The subdivision strategy relies on a simple heuristic: most human-constructed solid models are predominantly aligned with the axes of the coordinate system in which they were constructed (their 'local' coordinate system). We therefore measure the extent of the body in each of the local coordinate-axis directions, using vertex data, and choose to cut the body in half along the longest. (A more reliable algorithm is described in [8].)

Several improvements are possible to this algorithm, both in terms of the quality of the resulting sphere hierarchy and the speed of the construction process. We have found that the construction can be accelerated substantially by exploiting symmetry. If the two pieces generated by a cut pass a mirror symmetry test (assuming that the modeller supports this function) then we only construct a hierarchy for one piece, and use its mirror image for the other.

Many of the objects that are represented by sphere hierarchies will move. Whenever this occurs, it is necessary to recalculate the locations of the sphere centres. However, as there can be thousands of spheres in a hierarchy, we implement a lazy-evaluation scheme based on a reference position for the hierarchy and a 'current transform'.

The sphere hierarchy for a complete robot is constructed dynamically from its links in a bottom-up manner. Every time the robot moves, the links are transformed to their new positions and their hierarchies are placed on a list. The list is searched for the pair of hierarchies that can be enclosed in the smallest sphere. The two hierarchies are removed and combined, and the result is placed back on the list. The process repeats until the list length is 1. We use a fast, trivial algorithm to combine two hierarchies, that examines only the two root nodes.

3 Measures of Efficiency

The purpose of a sphere hierarchy is to define a volume that approximates the shape of a given body; and to provide an efficient means of classifying points as to whether they are inside or outside this volume. We say that the former are 'inside the hierarchy' and the latter 'outside the hierarchy'; and we define the efficiency of a sphere hierarchy to be the average cost of making this classification. (A similar idea appears in [14].)

The hierarchies we use are actually capable of a 3-way classification: points definitely inside (V_{in}), definitely outside (V_{out}), or possibly inside (V_{maybe}) the body represented by the hierarchy. The points in V_{maybe} are those that lie outside all inner spheres, but inside at least one outer sphere at a leaf node. For the purpose of efficiency measurement only, we simplify this to a 2-way classification between $V_{out} \cup V_{maybe}$ and V_{in}, on the grounds that it has the same cost as the 3-way classification.

A quantitative measure of efficiency can be defined as follows:

$$e(h, a, V) = \frac{1}{S} \int_V (c(\mathbf{p}, h, a) - 1) d\mathbf{p}$$

where e is the efficiency measure for sphere hierarchy h, a is a point-classification algorithm, $c(\mathbf{p}, h, a)$ is the cost of classifying point $\mathbf{p}$ as being inside or outside h via algorithm a, the cost being measured as the number of point-in-sphere tests performed, V is the volume over which the integration is performed, and S is the volume of the smallest sphere that will enclose the body represented by h. The purpose of S is to make e scale-invariant and dimensionless. We assume that a is a tree-search algorithm that visits the children of a given node only if the point is inside the outer sphere at that node.

V must wholly contain the enclosing sphere of h. However, as $c(\mathbf{p}, h, a) = 1$ for any point that is outside this sphere, there is no contribution to e from these points; so $e(h, a, V) = e(h, a, V \cap B)$ for any bounding volume B that wholly contains the enclosing sphere. Given that the outer bound of V is essentially arbitrary, there are only three intrinsically different choices for V: the whole of Euclidean space, the set of points exterior to the body (according to its true shape) and the set of points that are outside the sphere hierarchy. Call them V_1, V_2 and V_3, and let $e_1(h, a) = e(h, a, V_1)$, $e_2(h, a) = e(h, a, V_2)$ and $e_3(h) = e(h, a, V_3)$. Notice that e_3 is independent of a. It is also easier to analyse, giving rise to much simpler expressions than e_1 or e_2; so we choose e_3 for the branching factor analysis below.

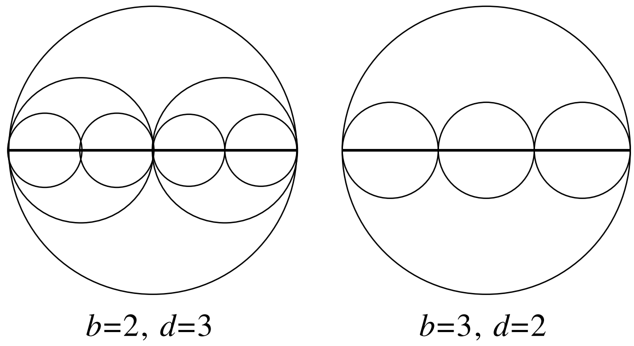

Figure 2: Sphere hierarchies covering a thin, straight rod.

Although e_1 and e_2 both depend on the choice of algorithm, the magnitude of the dependency is small. Furthermore, trials of various algorithms showed that depth-first search is the most efficient, as well as being the simplest to implement; so there is a good case to standardize on this algorithm for use in calculating e_1 and e_2.

4 Branching Factor Analysis

The purpose of this section is to present an analytical branching factor analysis. By its nature, such an analysis will be limited to certain types of shape. We will consider only rod-like and plate-like shapes. The alternative is of course to perform a numerical branching factor analysis. Such an analysis is applicable to any shape, and to any design of hierarchy, but the results are difficult to generalize.

Consider a sphere hierarchy with branching factor b and depth d covering a straight, zero-thickness rod. There is only one sensible design for such a hierarchy: cut the rod into b equal segments, enclose each one in a sphere, and recurse. A couple of examples are shown in Figure 2. The efficiency of this hierarchy is

$$e_3(h(b, d)) \;=\; \sum_{i=1}^{d-1} \frac{b^i}{b^{3(i-1)}} \;=\; \frac{b^3 - b^{5-2d}}{b^2 - 1}.$$

This equation is obtained by partitioning V_3 into components V_{3i}, $i = 0 \ldots d$, where V_{30} is the volume outside the enclosing sphere of h and V_{3i} is the union of all the (outer) spheres at level i minus the union of all the spheres at level $i+1$. (The root node is level 1.) The cost of classifying a point $\mathbf{p} \in V_{3i}$ is $1 + \min(i, d-1)\, b$, so

$$e_3(h(b,d)) \;=\; \frac{1}{S}\sum_{i=1}^{d} \min(i, d-1) \times b \times \mathrm{vol}(V_{3i}).$$

e_3 is not designed to penalise a hierarchy for lack of accuracy; so, to get a fair comparison, we must arrange for all the hierarchies to have the same accuracy. The easiest way to do this is to let $d \to \infty$, since $h(b, \infty)$ is exact for all $b > 1$. The efficiency expression then simplifies to $b^3/(b^2-1)$, which has a minimum value at $b = \sqrt{3}$. The nearest integer value is $b = 2$, which is therefore the optimum branching factor.

If the above analysis is repeated for a circle hierarchy covering a straight line segment in 2D space then the efficiency is

$$e_3(h(b,d)) \;=\; \frac{b^2 - b^{3-d}}{b-1},$$

and the optimal value of b is exactly 2. Notice that efficiency measures are sensitive to the dimension of the underlying space.

Strictly speaking, the above analysis applies only to an infinite-depth sphere hierarchy covering an exactly-straight rod of zero thickness. However, given that e_3 varies continuously with small changes in rod geometry, and that $e_3(h(b,d))$ converges as $d \to \infty$, we may conclude that $b = 2$ is the optimal branching factor for any finite-depth hierarchy approximating any shape that resembles a nearly-straight rod, provided that both the leaf-node spheres and the rod's thickness are substantially smaller than the rod's length.

Now let us consider some sphere hierarchies covering a zero-thickness flat plate. There is more than one way to design such a hierarchy, so we present results for each of the designs shown in Figure 3. Although each design is intended for a specific branching factor, it is still possible to parameterize their efficiencies in terms of b in a sensible manner. We use the following substitutions: the number of children to each parent is set equal to b, and the ratio of child sphere radius to parent sphere radius is $1/\sqrt{b}$. The efficiency of the rectangular covering is then

$$e_3(h(b,d)) \;=\; (1-C)b + C\,b\,\frac{1-(1/\sqrt{b})^{d-1}}{1-(1/\sqrt{b})},$$

and the efficiency of both the square and triangular coverings is

$$e_3(h(b,d)) \;=\; b\,\frac{1-(1/\sqrt{b})^{d-1}}{1-(1/\sqrt{b})}.$$

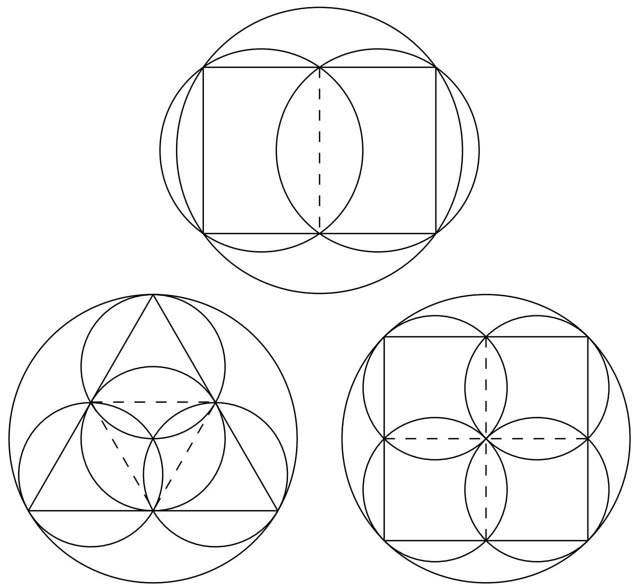

Figure 3: Sphere hierarchies covering a thin, flat plate.

The constant C is the proportion of the volume of a child sphere in the rectangular covering that is inside its parent. Its exact value is $C = 0.5+\sqrt{2}-5/(3\sqrt{3}) \simeq 0.95$, which means that the parameterized efficiency formulas differ from each other by less than 5%. Letting $d \to \infty$ and minimizing with respect to b produces the following optimal values: $b \simeq 2.23$ for the rectangular covering and $b = 2.25$ for the other two. In all cases, the optimal branching factor is 2.

To summarize, we have shown that *in general*, the optimal branching factor is 2 for each of the shapes that we have considered, which suggests that designers should aim for a branching factor of 2 whenever it is reasonable to do so. We do not claim that 2 is the optimal branching factor for every shape, because that is clearly false.

5 Results

We have implemented the algorithm described in Section 2 on a Silicon Graphics Indigo[2]; and we have interfaced it to two CAD systems: Robcad (via the ROSE API) and the ACIS geometric modeller. We tested the algorithm on a model of a commercial robot (IGM RT280), and also on a model of a desk-lamp. A picture of the desk-lamp produced by Robcad is shown in Figure 4, and a picture of the robot is included in Figure 6. The maximum reach of the robot is over 2m, while the desk-lamp can be stretched to about 1m long. The times quoted below refer to the total

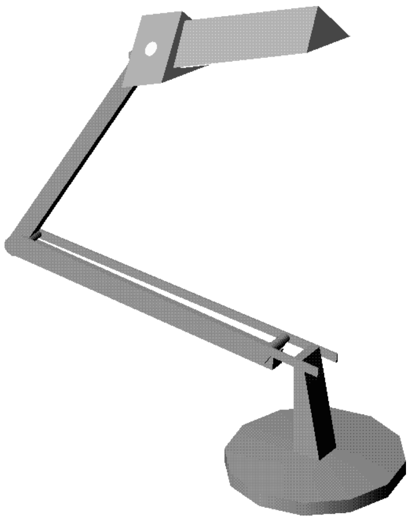

Figure 4: A polyhedral model of a desk-lamp.

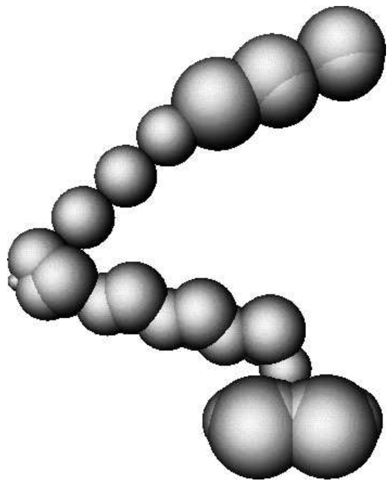

Figure 5: The leaf nodes of a sphere hierarchy which represents the desk-lamp. The level of accuracy used was 60mm (that is the maximum difference between the concentric radii of the leaf spheres).

elapsed time for a hierarchy to be generated, including the time taken by the geometric modeller (Robcad). Symmetry exploitation was not possible with Robcad, so the times exclude the effect of this particular optimization.

Sphere hierarchies for the desk-lamp were produced up to an accuracy of 2mm. (Recall that the accuracy of a sphere hierarchy is defined to be the maximum difference between the outer and inner radii of the leaf nodes). Since the lamp's geometry is relatively simple, most of the link sphere hierarchies could be constructed in a matter of seconds. The base of the desk-lamp is approximated by a cylinder and a frustum, and the algorithm proved to be slow at subdividing the frustum. Combining the link hierarchies took an unmeasurably small amount of time. The complete sphere hierarchy for the whole desk-lamp was produced in under a minute, and contained a total of nearly 50,000 spheres (for 2mm accuracy). The spheres shown in Figure 5 have a coarser accuracy (60mm) for clarity. The figure shows the 43 leaf spheres of the complete lamp hierarchy which contains a total of 85 spheres.

The robot proved to be more difficult to process. The complete sphere hierarchy was produced in about 5 minutes. To speed up the construction it was found that cuboids could be decomposed as special cases without using any of the usual geometric functions. As the construction of a hierarchy progresses with subdivisions in the Cartesian directions, the bodies produced are often cuboids, and therefore, solids with 8 vertices are tested to see if they are axis-aligned cuboids.

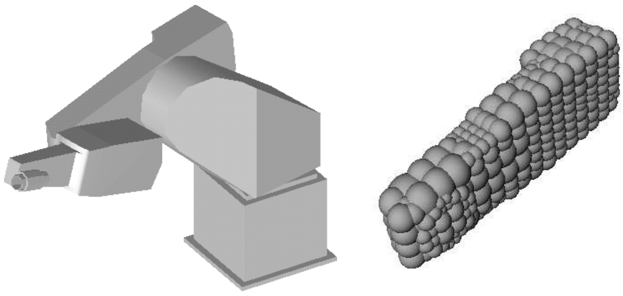

Figure 6: The model of the robot together with the leaves of the sphere hierarchy of one of the links (to an accuracy of 25mm).

The complete model of the robot is shown in Figure 6 together with the bottom level of one of the sphere hierarchies that was produced from it. The accuracy of these leaf spheres is 25mm.

Once constructed, sphere hierarchies are saved to disk so they can be read when they are needed in an intersection or collision test. Collision tests are performed recursively, and geometrical transformations are only applied where necessary. If the outer spheres of two sphere hierarchy nodes do not intersect then it is clear that they represent non-intersecting fragments and therefore their children need not be investigated. If the inner spheres of two nodes *do* intersect then a collision has been found and the calculation can be terminated. If neither of these cases occur and the nodes are not leaf nodes then a decision is made to investigate the children of one of the two nodes.

The minimum distance calculation is similar to the collision detection algorithm. The algorithm begins by calculating the distance between the root nodes as an interval consisting of the minimum (between outer spheres) and a maximum (between inner spheres). This interval is refined by examining the children of one of the nodes. The recursion can be pruned at any stage of the calculation if the pair are guaranteed to be further apart than the best interval found, or if the interval is accurate to some given tolerance. The latter condition is useful when large distances do not need to be known to a great accuracy, (see [13]).

6 Conclusion

An algorithm for constructing sphere hierarchies from geometric models of arbitrary polyhedral solids has been presented. The current implementation can interface with both Robcad and ACIS. It requests only the following data and operations of the geometry engine: the list of vertices, the list of polygon plane equations, the minimum distance from a point to the solid, and the operation of intersecting a solid with a planar half space.

Accurate sphere hierarchies can be synthesized in about 5 minutes for moderately complicated structures, like an industrial robot. Various optimizations are possible, including the exploitation of symmetry and detecting the cuboid special case.

A quantitative measure of the efficiency of a sphere hierarchy has been defined; and it has been applied to the problem of determining the optimal branching factor for a sphere hierarchy. Analytical results are presented for shapes that are basically rod-like or plate-like, showing that the optimal branching factor is 2. No analysis was presented for compact shapes. We have incorporated this result by adopting a binary-tree structure for our sphere hierarchies.

References

[1] Badler, N. I., O'Rourke, J. and Toltzis, H., A Spherical Representation of a Human Body for Visualizing Movement, *Proc. IEEE*, vol. 67, no. 10, pp. 1397–1403, 1979.

[2] Cameron, S., Efficient Intersection Tests for Objects Defined Constructively, *Int. Jnl. Robotics Research*, vol. 8, no. 1, pp. 3–25, 1989.

[3] del Pobil, A. P. and Serna, M. A., *Spatial Representation and Motion Planning*, Springer-Verlag, Berlin/-Heidelberg/New York, 1995.

[4] del Pobil, A. P., Pérez, M. and Martínez, B., A Practical Approach to Collision Detection between General Objects, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 779–784, Minneapolis, MN, 1996.

[5] Faverjon, B., Hierarchical Object Models for Efficient Anti-Collision Algorithms, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 333–340, Scottsdale, AZ, 1989.

[6] Featherstone, R., Swept Bubbles: A Method of Representing Swept Volume and Space Occupancy, Document no. MS-90-069, Philips Labs, Briarcliff Manor, NY, 1990.

[7] Featherstone, R., A Hierarchical Representation of the Space Occupancy of a Robot Mechanism, in J.-P. Merlet & B. Ravani (editors) *Computational Kinematics '95*, pp. 183–192, Kluwer Academic Publishers, Dordrecht/Boston/London, 1995.

[8] Gottschalk, S., Lin, M. C. and Manocha, D., OBBTree: A Hierarchical Structure for Rapid Interference Detection, *Proc. ACM Siggraph*, pp. 171–180, 1996.

[9] Greenspan, M. and Burtnyk, N., Obstacle Count Independent Real-Time Collision Avoidance, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1073–1080, Minneapolis, MN, 1996.

[10] Hubbard, P. M., Approximating Polyhedra with Spheres for Time-Critical Collision Detection, *ACM Trans. Graphics*, vol. 15, no. 3, pp. 179–210, 1996.

[11] Liu, Y., Noborio, H. and Arimoto, S., Hierarchical Sphere Model (HSM) and its Application for Checking an Interference between Moving Robots, *Proc. IEEE Int. Workshop on Intelligent Robots and Systems (IROS '88)*, pp. 801–806, Tokyo, 1988.

[12] O'Rourke, J. and Badler, N., Decomposition of Three-Dimensional Objects into Spheres, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 295–305, 1979.

[13] Quinlan, S., Efficient Distance Computation between Non-Convex Objects, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 3324–3329, San Diego, CA, 1994.

[14] Sawatzky, G. and El-Zorkany, H., Using an Efficient Collision Detector in the Solution of the Find-Path Problem of Industrial Robots, *Proc. SPIE, vol. 579, Intelligent Robots and Computer Vision*, pp. 131–141, 1985.

[15] Stark, J. S. H., Method for Preventing Collision of Two Mutually Movable Bodies and an Apparatus Including an Arrangement for Preventing Collision, US Patent no. 4,578,757, March 1986.

[16] Verwer, B. J. H., A Multiresolution Work Space, Multiresolution Configuration Space Approach to Solve the Path Planning Problem, *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2107–2112, Cincinnati, 1990.