Fast Distance Computation for On-line Collision Detection with Multi-Arm Robots

Dominik Henrich and Xiaoqing Cheng

Institute for Real-Time Computer Systems and Robotics
Prof. Dr.-Ing. U. Rembold, Prof. Dr.-Ing. R. Dillmann
University of Karlsruhe, Kaiserstr. 12, 7500 Karlsruhe, F.R.G.

Abstract

For the on-line collision detection with a multi-arm robot a fast method for computing the so-called collision vector is presented. Manipulators and obstacles are modelled by sets of convex polytopes. Known distance algorithms serve as a foundation. To speed up the collision detection dynamic obstacles are approximated by geometric primitives and organized in hierarchies. On-line, the here introduced Dynamic Hierarchies are adjusted to the current arm configuration. A comparison with previous methods shows an increased acceleration of the computations.

1 Introduction

A robot with two independent arms in a hanging configuration gave the impulse for this work. Both manipulators have six rotational degrees of freedom in overlapping working spaces. Only by the cooperation of two arms, large classes of difficult manipulation tasks can be carried out. As a presupposition therefore one must secure, that no collisions with the two arms mutually as well as with other objects existing in the environment come about.

A frequent approach to collision detection is the transformation of the objects in the Configuration Space, chosen among the spaces of independent parameters describing the position of any point bound to the arm. The robot shrinks to a point by enlarging the obstacles. Problems arise with moving objects, which must again be transformed to the Configuration Space after every movement. On-line that is impossible.

Other approaches work in the Cartesian space. Thereby a collision is detected by intersections between the geometry of obstacles and robot. Beside of the pure test on collision, here the so-called collision vector is calculated. The collision vector states the minimal distance and its orientation between the robot arm and other possibly moving objects. By the additional distance and orientation information the following collision avoidance can be improved.

Here a fast computation of the collision vector for the on-line collision detection with several robot arms is presented. It is based on a hierarchical modelling of dynamic obstacles.

Previous approaches are sketched and classified in Sec. 2. For our own approach in Sec. 3 a formal world model is introduced. The Sec. 3.1 discusses the approximation by primitives. The new Dynamic Hierarchies are presented in Sec. 3.2. The different approaches are compared in Sec. 4.

2 Previous approaches

Most other researchers working on fast computation of the collision vector use a combination of several approaches. Instead of presenting the full combinations, here the underlying approaches for speed up are separately described. For that purpose the straightforward computation of the collision vector is given:

During the execution of a continuous movement of the manipulator the collision vector is calculated at discrete moments. For this the collision vector is computed among all pairs of objects, each consisting of an arm segment and an obstacle. The shortest of these vectors is reported as the result of the collision detection.

This straightforward approach can be accelerated under four aspects. They differ by the method and the point of time. The point of time, when the speed up is contrived, can be off-line or on-line. The method can be on one hand a reduction of the amount of considered objects and on the other hand an improvement of a single distance computation. If the point of time and the method are combined, the four different aspects of acceleration develop. In Tab. 1 for every aspect an example is given, which shall be sketched in the following:

Primitive Approximation: The obstacles or the arm segments can be approximated off-line by one or more simple geometric objects (primitives). See for example Adolphs and Horsch in [1]. By this means the complexity of a single distance computation is reduced. The collision vector is an approximation of the real vector. Because of the coarser representation possibly a feasible path seems unfeasible

Initialization: During the path planning a single computation of the collision vector can be improved by using the result of the last motion step. For bodies modelled by convex polytopes this was done by Gilbert and Johnson in an iterative distance calculation in [5]. The vector from the last step serves as an initialization for the new iteration. The degree of improvement depends on the covered distance by one motion step.

Distance Update: In this approach a discrete trajectory is assumed. The furthest distance covered by the robot arm is

subtracted from the length of the last collision vector. If the result is smaller then a threshold, then the vector is newly calculated. See Faverjon in [3]. For example, this update of the distance can be done for every object pair (locally) or for only the smallest vector (globally). By this the amount of computations on a trajectory is diminished. Nevertheless the direction information of the collision vector gets lost.

Static Hierarchies: Before path planning, previous obstacles are successively combined and approximated into new obstacles. This develops a hierarchical arrangement of approximations. This so-called assembly-tree was used by Faverjon in [2]. At each level of the hierarchy all obstacles are represented completely. The higher the level, the coarser the representation. Thus the amount of obstacles are reduced logarithmically. Because of their dynamic behavior entire manipulators can´t be approximated by Static Hierarchies (see Sec. 3.2). Only single arm segments can be represented in this way.

Virtual Manipulator: To reduce the amount of arm segments a sequence of virtual manipulators with different degrees of freedom can be computed off-line. According to Faverjon in [3] a *virtual manipulator* with k degrees of freedom is that manipulator, whose last links have been replaced by their swept volumes, when joints $k+1$ to n are made free. Here the original manipulator has n degrees of freedom. A virtual manipulator represents a relative coarse approximation of its successor.

	off-line	on-line
quantity of computations	Static Hierarchies; Virtual Manipulator	Distance Update
complexity of one computation	Primitive Approximation	Initialization

Tab. 1: Examples for acceleration aspects.

3 Our approach

To describe our approach, first a formal world model is introduced. Then, in the following subsections the two main aspects of our approach are discussed: the Primitive Approximation and the Dynamic Hierarchies.

Definition of the collision classes

In a known environment all geometric objects are collected in a world model. This world model usually is divided for the purpose of collision detection in dynamic arm segments of a robot and static obstacles touchable by the robot. Because collision detection is being done between more than one arm, we subdivide the world model into collision classes.

The elements of a collision class are characterized by the fact, that no collision detection is done between them. A collision class either contains objects where no collisions can occur between them or their collisions are easier avoided by other mechanisms. For example one could form a collision class from the segments of a manipulator. Then the joint angles must be checked to avoid internal collisi-

ons. Other collision classes could be formed by the obstacles in the environment. The obstacles are not moveable, so no collision can occur among them.

This classification is constant, that means the membership of an object to its collision class stays unchanged. Nevertheless the location or the geometry of the objects can change over time. The word model contains the geometric information of the scene necessary for the collision detection. The real objects are modelled by convex polytopes. The *world model W* consists of a set of collision classes C^j:

$$W := \left\{ C^j \mid j = 1,\dots,c \right\}.$$

It is defined in a given world frame F_w . In a *collision class* C^j the real objects are modelled as a set of convex polytopes P_i^j :

$$C^j := \left\{ P_i^j \mid i = 1,\dots,N^j \right\}.$$

Relative to the word frame F_w each collision class C^j has got its own local frame F^j. Usually the *convex polytopes P* are described by the convex hull of the vertices R of the real object:

$$P := \text{co}(R).$$

Additionally every polytope P_i^j has got a local frame F_i^j, relative to F^j.

Specification of the collision detection

At a fixed point of time t the collision detection computes for a given world model W the *collision vector* $v^{j,l}$ between two collision classes

$$C^j := \left\{ P_i^j \mid i = 1,\dots,N^j \right\},$$

$$C^l := \left\{ P_k^l \mid k = 1,\dots,N^l \right\},$$

with $j,l \in \{1,\dots,c\}$, $j \neq l$ by:

$$v^{j,l} := v_{i,k}^{j,l} ,$$

for $\left| v_{i,k}^{j,l} \right|$ = min, with $i \in \{1,\dots,N^j\}$, $k \in \{1,\dots,N^l\}$. Generally, the collision vector $v_{i,k}^{j,l}$ between the two convex polytopes $P_i^j \in C^j$ and $P_k^l \in C^l$ is defined in the following way:

$$v_{i,k}^{j,l} := r - s,$$

for $|r\text{-}s|$ = min, with $r \in P_i^j$ and $s \in P_k^l$. The Euclidean Norm in the 3-dimensional space is denoted by $|\cdot|$.

Additionally an index set I is defined for the whole world model:

$$I := \left\{ (j,l) \mid j \neq l, j,l \in \{1,\dots,c\} \right\}.$$

It states for each unordered pair $(j,l) \in I$ between which classes C^j, C^l collisions should be detected.

The usefulness of the index set appears at the following constellation: Two manipulators are mounted in a hanging configuration on a vehicle via a gallows. Arm, gallows and vehicle form each a class: $C^1,...,C^4$. In this index set all of the pairs are entered except of arm/gallows (1,3),(2,3) and gallows/vehicle (3,4). Otherwise these pairs always would produce a collision. This index set specifies only the pairs of classes, where collision detection is reasonable:

$$I = \{(1,2), (1,4), (2,4), (3,4)\}.$$

With the declaration of collision classes in a world model and a corresponding index set the task of the collision detection is completely specified.

Properties of the collision classes

If collision classes are inspected over the planning time, diverse properties can be captured. The diverse properties result from the movement of objects in a class. The objects can perform together a synchronous movement or change their position relative to each other. A collision class C^j is called *fixed* over the time t, if the class frame

$$F^j(t) = \text{const},$$

otherwise C^j is called *moved*. At a *static* collision class C^j over the time t the polytope frames are

$$F_i^j(t) = \text{const for } i = 1,...,N^j,$$

otherwise C^j is called *dynamic*. Examples for classes with this four properties are listed in Tab. 2.

property	static	dynamic
fixed	table	manipulator
moved	vehicle	mobile robot

Tab. 2: Examples of collision classes

3.1 Primitive approximation

Modelling with primitives

Following the world model objects are modelled by convex polytopes. To speed up the computation of the collision vector, the convex polytopes can off-line be approximated by primitives. By a *primitive*, a simple convex volume is understood. These approximations are conservative, i. e. the polytope volume is completely covered by the primitive. There exists a relation of partial sets between convex polytope P and its approximation A_i:

$$A_i \supseteq P.$$

Adolphs and Horsch in [1] used as primitives: box, cylinder, sphere and bounding-box. Thereby the cylinder is a spherical extension of a line segment in space (describable by the two end points and the radius). The bounding-box is a more simple box with faces parallel to the frame axes. In Fig. 1 an arm segment with its Primitive Approximation is shown.

The order of the primitives as approximations essentially corresponds to the complexity of the primitives, measured by the minimal amount of scalar parameters needed to describe the primitive. For example the sphere needs: for the mid point 3 and for the radius 1 parameter. If it is assumed, that in the most cases objects are better approximated by more complex primitives than by less complex, then the original object is represented by the primitive approximation in different levels of accuracy.
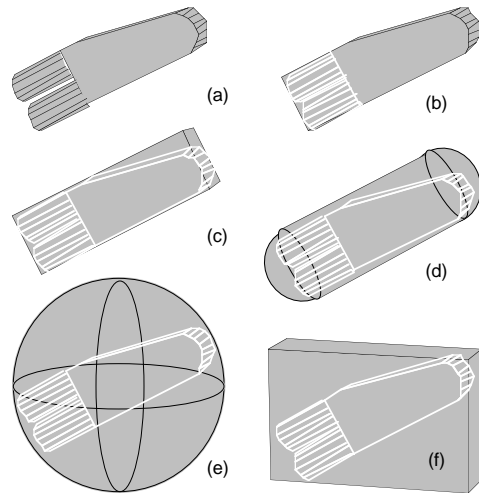

Fig. 1: Primitive Approximation of an arm segment; (a): real object; (b): convex polytope; (c): box; (d): cylinder; (e): sphere; (f): bounding-box.

Distance computation between objects

The distance computation of the collision detection can be sped up by the Primitive Approximation calculated off-line by the convex polytopes. For this the collision vector is computed among the primitives, instead of among the convex polytopes. Because the distance between two primitives is never greater than between the approximated polytopes, it represents a reasonable approach to the real distance.

The distance computation between two primitives of the same kind can take place by the algorithms of Gilbert and Johnson [5], Meyer [9] and Lumelsky [8]. Between different primitives the algorithms are combined. While computing the collision vector between two objects, in the first step a pair of primitives approximating the objects are selected by a certain strategy. Then the collision vector is computed between the selected pair. If the distance is greater than a given threshold, then the vector can be returned as the result. If it is smaller and there exists still another unused pair of primitives, then these steps are repeated. When computation is not successful with the primitives to a distance greater than the threshold, then the vector is calculated between the convex polytopes.

For selecting the primitive pairs, for example, the following strategies can be applied:

- sort the pairs by the complexity of the distance computation;

- by selecting the appropriate pair reducing the approximation error (for example calculate the volume difference between primitive and object);
- choose for every object the next primitive by a fixed order;
- for both objects select as first primitive the bounding-box and as the next primitive the convex polytope. Leave the rest.

Experiments have shown, that the last strategy in selecting the primitive pairs is the the fastest method for distance computation. Except for bounding-box and the convex polytope the other primitives slow down the computation. The bounding-box serves as a quick pretest. For details see Henrich in [6].

3.2 Dynamic Hierarchies

Hierarchical modelling

To reduce the amount of members in a class for collision detection their objects can be combined and approximated by compositions. *Compositions* are conservative approximations of several objects by one primitive. Replacing these objects of a collision class successively by their compositions, so a hierarchy of compositions is build up. It is similar to the assembly-tree of Faverjon in [3]. The elements of one level of the hierarchy approximate the original objects of the collision class in a certain granularity. The higher the level, the coarser the representation of the objects. At the lowest hierarchy level all the original objects are settled, at the topmost level there is only one composition. But yet it contains all the objects of the class.

Constructing optimal hierarchies

In general a lot of different composition hierarchies could be build up for one collision class. One hierarchy is said to be good if it enables a fast computation of the collision vector. The idea of the Dynamic Hierarchies is to construct optimal hierarchies which reflect the objects constellation of a collision class at every moment. Descending in such an optimal hierarchy the distance computation can quickly find out the relevant object which contains one end of the collision vector. At a particular time a hierarchy of compositions is called *optimal* for a collision class if a criterion function is minimized at each level of this hierarchy.

When constructing optimal hierarchies the following criterion functions can be used to compose elements of a hierarchy level to elements of the next higher one:
- volume of the composition,
- diameter of the composition and
- surface of the composition.

The diameter of the composition turned out to be the best of the three criteria. It can be computed quite fast and develops hierarchy structures corresponding to those constructed by hand.

For a given collision class and a criterion function an optimal composition hierarchy can be off-line computed. If

the criterion function satisfies the triangle inequality the Nearest Neighbour Algorithm [2] can be used:

At first, all class members are inserted into a list. Then this pair of objects is searched in the list, which minimizes the criterion function. For this pair a composition is computed and the two original objects are replaced by this composition. Now, the list is shorter by one element. These steps are repeated as long as the list has more than one element. The result is a optimal binary tree with respect to the criterion function. Fig. 2 shows two examples of such optimal hierarchies.

The linear progression of the algorithm should not deceive one about, that for the most classes it is a matter of a logarithmical reduction of the objects. Generally, pathological object constellations of a collision class can give rise to only a linear reduction. However it is not due to the algorithm, but in the constellation of the objects or in the criterion function.

For static classes ($F_i^j(t)$=const for all i) the optimal hierarchies can off-line be computed by the above-mentioned algorithm and directly applied for on-line collision detection. For dynamic collision classes the possible alterations of the objects relative to each other are known in advance. With these classes for each configuration an optimal composition hierarchy can off-line be computed.

Using a robot arm one would specify the intervals of joint angles in which one tree structure of the composition hierarchy remains optimal. For a Puma260 arm with a sampling rate of 15 degrees for the first three joints 19 composition hierarchies were computed. For two thirds of the arm configurations one and only one tree structure is optimal (see Fig. 2a). The 18 remaining tree structures are adopted only in extreme arm positions (see Fig. 2b).
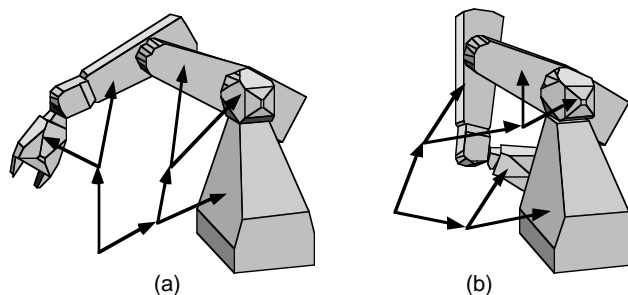

(a)   (b)

Fig. 2: Two different optimal hierarchy structures for a Puma260 depending on the arm configuration

Thus for the Puma260 just one composition hierarchy is actually necessary. But for a general manipulator more than one optimal hierarchies should be used. How to select the optimal hierarchy for the current arm configuration is shown in the following.

Selecting the optimal hierarchy structure

During the on-line collision detection for a given dynamic collision class the current optimal hierarchy structure must be selected among all the possibly optimal structures constructed off-line.

Given the number of objects of a collision class N, the total number of the in theory possible optimal hierarchy structures grows exponentially by N. Because the optimal structures are constructed basing on the possible alterations of the objects relative to each other within a collision class, many of the theoretically possible hierarchy structures do not occur. Additionally, if this set of optimal hierarchy structures is still too large for implementing an efficient selection function, the less often used structures can be cut off. So the optimal structure can be searched within a reasonable amount of off-line constructed optimal hierarchy structures.

For a dynamic collision class the above-mentioned criterion function is evaluated for the compositions of the lowest level of all the possibly optimal hierarchies. According to the properties of the Nearest Neighbour Algorithm the hierarchy structure which minimizes the criterion function at the lowest level of the hierarchies is the current optimal hierarchy. If the criterion function value is minimal for more than one hierarchy the comparison is repeated at the next higher level of these hierarchies.

Because the criterion function can be computed quite fast and the amount of the off-line constructed optimal hierarchy structures remains in reasonable range this selection method is practicable for the on-line collision detection.

Update of the hierarchy compositions

Having selected the optimal hierarchy structure all the compositions of this hierarchy must be calculated now. While specifying the composition hierarchy the choice of the primitive being used for composing is left open. Building up a hierarchy one has got the freedom to use for example, convex polytopes or simple primitives as compositions.
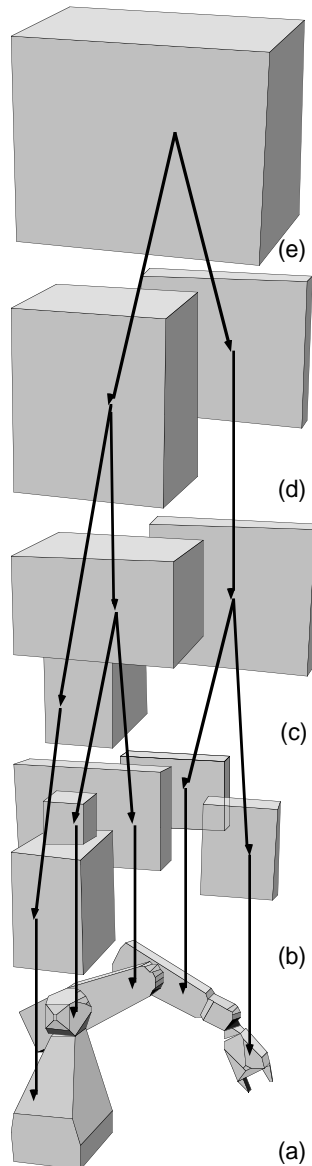

Fig. 3: Modelling of a Puma260 arm as Dynamic Hierarchy; (a): convex polytopes; (b): bounding-box approximation; (b)-(e): bounding-box hierarchy.

From the results of the primitive approximation (in Sec. 3.1) the bounding-box in combination with the convex polytopes proves as the best primitive for computing the collision vector. Moreover an approximation through a bounding-box can be computed very fast on-line. Therefore the bounding-box is chosen as primitive for a composition hierarchy. See Figure 3(b)-(e).

After every alteration in a collision class (with manipulators after every motion step) all compositions in the hierarchy are on-line again computed.

Summary of the modelling

Summarizing, a combination of primitive approximation and composition hierarchies are suggested for modelling (static and dynamic) collision classes. The real objects of a class are modelled by convex polytopes (Fig. 3a) and approximated by bounding-boxes (Fig. 3b). Above that a hierarchical approximation out of bounding-boxes, with dynamical tree-structure and geometry, is build up (Fig. 3b-e).

This changing hierarchy with selected optimal tree-structure and updated geometry for every movement is called *Dynamic Hierarchy*. The Dynamic Hierarchies cover all four aspects of acceleration classified in Sec. 2.

Descending in hierarchies

During collision detection between two collision classes represented by Dynamic Hierarchies an approximation of the collision vector shall be computed. Therefore the two topmost hierarchy levels containing only one composition will be considered. At each level among all the composition pairs the collision vector is computed. If the distance between a pair is smaller than the threshold, then following a given strategy the members of that pair are substituted by their more precise representation at the level underneath. With these new pairs the computation is continued. The recursive algorithm terminates, if either the distance among all viewed pairs is greater than the threshold or one pair at the lowest (most accurate) representation level of both classes are colliding.

In a pair, where the distance is to small, one partner must be selected for substitution by different strategies. For example:

- substitute only *one* member of the pair: choose the one with the greater approximation error;
- substitute *both* members of the pair by their more accurate representation.

Here the latter strategy is used, because it does not require any computations and it descends most quickly in the hierarchy.

4 Experimental results

In two simulated examples, previous and the presented approaches are compared. The first contains a Puma260 consisting of 10 parts and 16 single obstacles (Fig. 4a); the second contains two manipulators each with 10 segments (Fig. 4b). Thus in every scene two collision classes could be formed. The system was implemented on a

SUN/SPARC 2 under UNIX in the programming language C.

For one motion step of the manipulator the average run time of the computations of the collision vector between the two classes was measured. Thereby the run times were separated in steps with and without collision(s) (see Fig. 5).

Foundation for all the implemented approaches is the modelling of objects by convex polytopes and the distance computation by Gilbert and Johnson (Fig. 5(a)). Compared are the Dynamic Hierarchies in Fig. 5(e) with: the Initialization in Fig. 5(g) corresponding to the description in [5]; the global Distance Update in Fig. 5(c) without negative distances; and with the Primitive Approximation with a symmetrical descending in the four primitives of [1].
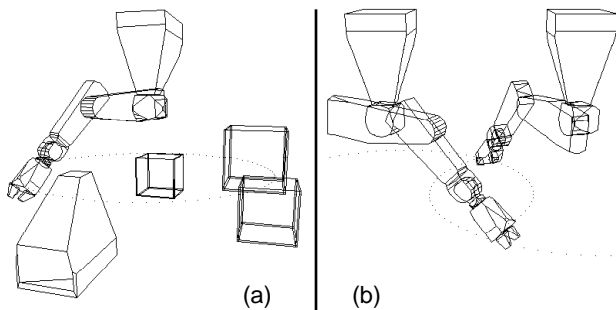

Fig. 4: Example scenes represented by convex polytopes.

Not recorded is the time for transformation of the convex polytopes in the world model after a dynamical change (arm movement). With a given modelling the transformation time is identical for all the approaches is consistently ca. 25 ms.

The comparison of the approaches shows the greatest acceleration at the Dynamic Hierarchies. A combination with the Initialization approach or the Local Distance Update deteriorates the run time, because of the data management necessary therefore. By a combination with the global Distance Update the information about the direction of the collision vector gets lost. With the Dynamic Hierarchies an on-line adjustment of the composition to the movements of a robot arm is possible.

The presented approach for the accelerated computation of the collision vector with multi-arm robots can be applied without any alteration as a fast collision *test*. This is needed for example in the off-line path planning. Based on the properties of the Nearest Neighbour Algorithm, the *precise* collision vector can be computed quickly by combining the Dynamic Hierarchies with an A*-search for further applications.
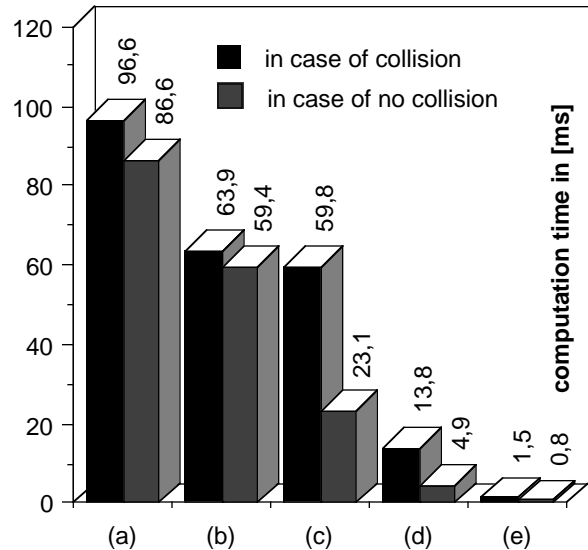
Acknowledgements

Fig. 5: Run-time of the acceleration approaches; (a): Convex Polytopes; (b): Initialization; (c): Global Distance Update; (d): Primitive Approximation; (e): Dynamic Hierarchies.

References

[1] Adolphs P., Horsch T., Olomski J., Höhn G.: CARO - ein Konzept für die kollisionsvermeidende on-line Bahnplanung für einen 6-achsigen Roboter. TH Darmstadt, Fachgebiet Regelsystemtheorie und Robotik, interner Bericht

[2] Duda R., Hart P.: Pattern classification and scene analysis. Wiley, New York, 1972

[3] Faverjon B.: Hierarchical object models for efficient anti-collision algorithms. IEEE, 1989

[4] Freund E., Hoyer H.: Real-time pathfinding in multirobot systems including obstacle avoidance. Int. Jour. of Rob. Res., Vol 7, No 1, S 42, Feb. 1988

[5] Gilbert E. G., Johnson D. W., Keerthi S.: A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal of Rob. and Autom., Vol 4, No 2, April 1988.

[6] Henrich D.: On-line Kollisionserkennung mit hierarchisch modellierten Hindernissen für ein Mehrarm-Robotersystem. Master´s Thesis, University of Karlsruhe, 1991

[7] Khatib O.: Real-time obstacle avoidance for manipulators and mobile robots. The Int. Jour. of Rob. Res., Vol 5, No 1, Spring 1986.

[8] Lumelsky V. J.: On fast computation of distance between line segments. Inform. Proc. Let., Vol 21, No 2, Aug. 1985, S 55

[9] Meyer W.: Distances between boxes: Application to collision detection and clipping. Proc. IEEE Int. Conf. Rob. Autom., S 587, 1986