Proceedings of DETC'01
ASME 2001 Design Engineering Technical Conference
and Computers and Information in Engineering Conference
Pittsburgh, PA, September 9-12, 2001

DETC2001/DAC-21057

A GENERAL METHOD FOR COMPUTING THE REACHABLE SPACE OF MECHANISMS

Artur Fuhrmann
DaimlerChrysler Research and Technology
Virtual Reality Competence Center (FT3EV)
Ulm, Germany
`artur.fuhrmann@daimlerchrysler.com`

Elmar Schömer
Max-Planck-Institute for Computer Science
Algorithms and Complexity Group
Saarbrücken, Germany
`elmar.schoemer@mpi-sb.mpg.de`

ABSTRACT

A new methodology[1] is presented for computing a minimal envelope for the reachable space of a mechanism, i.e. the space that contains a given mechanism in all its admissible configurations. The research is motivated by the packaging process in Digital Mock-Up applied in automotive industry. An important task in the concept phase, the automated determination of the space requirement for all parts, is still an unsolved problem in the case of mechanisms. The particular benefit of the method presented is its generality and robustness: It is able to deal with both open- and closed-loop mechanisms. The reachable space is computed with regard to the geometric description of each part. The approximation is enclosing and always converges in a uniform way and the tolerance can be pre-defined by the user. The method combines the use of bounding object hierarchies and the application of interval analysis. It is also able to approximate the swept volume of an object following a parameterized trajectory. We describe how the efficiency can be improved by lazy evaluation and by a deeper problem analysis. The presented algorithms are implemented and tested to a large extent[2].

1 Introduction

The changing markets require new products that have to be developed and produced with higher quality, in shorter periods and at a minimum of costs. For automotive and aerospace industry, *Digital Mock-Up (DMU)* is one of the key success factors for dealing with these challenges. It is often defined as a realistic computer simulation of a product, providing all required functionalities for design, manufacturing and maintenance. It serves as a basis of product development and influences important business decisions. Presently, the development process is still oriented to *Physical Mock-Ups (PMU)*, but the importance of DMU is steadily increasing. PMUs are still essential, but the use of DMU helps the engineers to save many PMUs and to increase their maturity.

A very important process as a part of the *digital validation* is the *packaging* process: The digital verification of the physical construction of the vehicle. Packaging can be seen as a well-defined process between component engineers and engineers responsible for the construction of the entire vehicle. Its objective is the determination of how space is occupied, to check parts for minimum clearance, in particular the avoidance of collisions, and also the investigation of the assembly and disassembly of objects. The results are digitally documented in the form of e.g. distances, geometric adjacencies between parts, envelopes for paths and geometric descriptions of occupied space. This information is often managed by the underlying *product data management (PDM)* system. Packaging relevant objects that complicate the work of packaging engineers are cables, hoses, clips and in particular mechanisms. These parts are dynamic objects and associated with a particular offset tolerance.

Static packaging, in contrary to its dynamic counterpart, is often seen as technologically simply. It mainly deals with the computation of the space requirements of objects and with static

[1] Patent pending (DaimlerChrysler AG, (Fuhrmann et al., 2000))
[2] This work will be part of the Ph.D. thesis of the first author and represents ongoing research.

collision checks. Indeed these problems are already solved in the case of rigid objects (*solids*). On the other hand, it is still a difficult and time-consuming task to determine the minimal space needed for a complex mechanism. It is even unclear how to define this space requirement of a mechanism. We mainly distinguish between two types:

Motion driven The *motion driven* space requirement is the space that a mechanism (e.g. a wheel suspension) needs for its most characteristic motions. These may be evaluated statistically by physical tests or simulations, and then are used to feed a swept volume simulation.

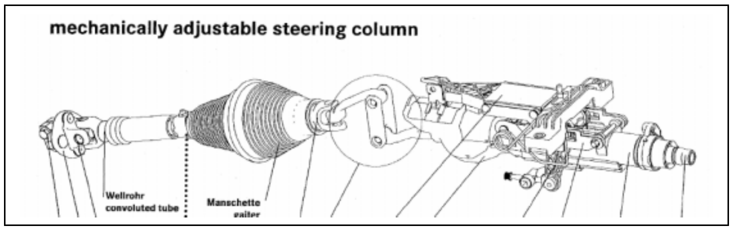
Reachable space The *reachable space (the (primary) workspace)* is the set of all points in space that can be reached by the mechanism within a legal configuration.

This paper deals with the latter problem. The goal is to present new methodologies that meet the demands of packaging engineers. The ranking of the main objectives is as follows.

1. *Generality and automation.* The problem should be solved in a very general way, no case distinction should be necessary. The computation should be able to run as a batch job over night.
2. *Accuracy and robustness.* The computation should be very accurate and robust. The accuracy should be freely definable.
3. *Efficiency.* The method should be able to deal with complex kinematic structures with many degrees of freedom and with the complex geometric shapes of the involved bodies.

An exemplary practical application is given by the steering column of an automobile. In order to be adjustable in height and depth, it consists of a number of revolute, prismatic and cardan joints. These form a kinematic chain that is connected to the car body through two further joints. The result is a closed-chain mechanism with many degrees of freedom for which the estimation of the space requirement represents a non-trivial task for the packaging engineer.

Presently the problem is solved by a trial-and-error approach with the aid of CAx tools. Based on the engineers' expert knowledge, the union of known *critical* configurations is computed, then the column is virtually rotated about its main axis and the *swept volume* of this rotation is calculated. This result is taken as an approximation for the space requirement, however nothing is known about the error made by this procedure. Obviously this approach has a strong dependence on specific properties of the mechanism. Up to what extent expert knowledge can be exploited, depends on the kinematic differences between a novel mechanism and its predecessor.



As far as known to the authors, no commercial software offers a functionality that is able to compute the reachable space in general. Packaging engineers often try to solve the problem according to the motion-driven case. To meet these demands, modern CAx systems offer various tools for computing swept volumes and envelopes. Indeed a procedure is known that recursively determines the reachable space by a sequence of swept volume and union operations[3]. However it is only applicable to open kinematic chains. A discrete variant of this approach is described e.g. in (Gupta, 1997). The computation of the swept volume is not a trivial task, see (Wang and Wang, 1986), (Blackmore et al., 1992) and (Abdel-Malek et al., 2000). Personal communications with packaging engineers reveal that the use of swept volume tools, even of modern CAx systems, results in an unmanageable amount of data, for the tools also sweep the interior of each part. As already indicated, this method is inherently connected with the property of open kinematic chains that their set of all admissible parameters is given by an interval[4]. However in the case of closed kinematic loops this set is a highly nonlinear manifold, so it is very unlikely that this method could be applied somehow.

In robotics, the space requirement of a mechanism is considered as the *reachable* or *total workspace,* which specifies the space that is reachable by the *tool-center-point* of the *end-effector* of a mechanical manipulator. Similar to approaches in the field of inverse kinematics, the investigation of workspaces in robotics strongly benefits from special properties of mechanical manipulators. Frequently additional assumptions are made about the degrees of freedom or about the type of joints. Structures with closed loops are not very intensively treated and the geometric shape of the parts is neglected in almost all cases.

Many methods are analytical or numerical in nature and often based on rank deficiency criteria of the position jacobian. See the works of e.g. (Kumar and Waldron, 1981), (Rastegar and Deravi, 1987). (Abdel-Malek et al., 1999) present methods for the determination of parametric equations of surface patches that envelop the workspace of serial manipulators. The equations are then evaluated with the aid of a mathematical package. This approach yields very accurate solutions and is well-

[3]In order to sketch this, we consider an open chain with joints $J_1, \dots, J_n$ connecting the bodies $B_0, \dots, B_n$, with B_0 denoting the base. In order to generate the workspace W_k for the joints $J_k, \dots, J_n$ and bodies $B_k, \dots, B_n$, we first unify W_{k+1} with B_k to a single solid. Then we compute the volume that is swept by this solid while joint J_k is moved through its entire range. The total workspace is given by the union of W_1 and B_0.

[4]We firstly neglect self collisions between components of the mechanism

suited for deeper analyses. But the authors also state that it is not capable of handling closed loops. The numerical method by (Haug, 1995) may also be used for the study of closed-loop manipulators. It is based on a highly nonlinear system derived from the jacobian rank deficiency criterion. The solution to the system describes inner and outer boundaries of the workspace and is traced by sophisticated continuation methods as curves on cutting planes. These global methods have to deal with special difficulties like bifurcation points and the finding of initial solutions of large-scale systems, see also (Wang and Wu, 1993). A comparison between the analytical and numerical method can be found in (Abdel-Malek et al., 1997). These types of methods are well-suited to compute the workspace for a single reference point. However it is unclear, how they can efficiently be applied[5] to solve the problem with regard to complex geometric shapes. The direct combination of the geometric representation of a body and the workspace boundary for its reference point does not yield the desired result. We address this again in subsection 2.2.

A voxel approach, discretizing the cartesian space and checking the reachablility of lattice vertices, could potentially solve the problem in general. However it leads to a very hard sub-problem: To decide in a globally correct way, whether any point of a kinematically restricted body is able to reach a given voxel. This problem has apparently not attracted any attention so far. We investigate this in (Fuhrmann, 2001). The global solution of the problem requires an expensive search in parameter space, hence we consider our method presented in section 4 to be more direct and more efficient.

In conclusion, there seems to be a lack of methods that are general enough for the purposes of packaging engineers. On one hand, geometrically based methods (e.g. swept volumes) are not able to handle closed kinematic chains, on the other hand, it is unclear from the viewpoint of numerical algorithms how to deal with a complex geometry efficiently.

2 Preliminaries

This section introduces the basic terminology and basic facts about mechanisms and kinematics and our main tools like bounding object hierarchies and interval mathematics.

2.1 Mechanisms

This paper deals with mechanisms in terms of multibody systems, a set of bodies (also called *links* or *parts*) interconnected by joints. The angle of each joint is given by a number of parameters ϕ_i which as a whole define the *parameter space* $D \subset \mathbb{R}^n$. This hyper-interval D is given by the joint ranges $\underline{D_i} \leq \phi_i \leq \overline{D_i}, i = 1, \ldots, n$. Every point in parameter space uniquely defines a *configuration* of the system, i.e. the position and orientation of each body.

The ”topological” structure of a mechanism can be described as a graph. We distinguish whether the graph contains cycles or not: if there are no cycles, the mechanism is an *open kinematic chain* or has a *tree structure*. In the case of *closed chain structures* we have the more complex situation that several joint parameters are no longer independent: the space of all parameters that keep the chains connected is implicitly defined by (in general highly nonlinear) constraints and forms a submanifold in D. Besides joint ranges and constraints, the set of admissible configurations can be further restricted by the prohibition of collisions between bodies. We restrict ourselves to the handling of closed-loop constraints and denote the set of these parameter tuples as $A \subseteq D$. We raise the issue of self collisions again in section 4.3.3.

The position and orientation of the reference point of each body in terms of system parameters defines the *forward kinematic*

$$\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^{4\times4};\ \mathbf{f}(\phi_1, \ldots, \phi_n) = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{1}$$

with the translational part $\mathbf{t} \in \mathbb{R}^3$ and the orthogonal matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ describing the rotation. We use such transformation matrices to simplify our description in this paper. In our implementation the rotational part is also described by quaternions where this is appropriate.

2.2 Reachable Space: Definition and Properties

Let a body B of a mechanical system be given as a compact set in $\mathbb{R}^3$ and let $\mathbf{f}$ be its forward kinematic as defined above. We denote the *reachable space* of B by $\mathcal{R}(B)$. The reachable space of a mechanism is the union of all reachable spaces of its bodies. A point $\mathbf{x} \in \mathbb{R}^3$ lies in $\mathcal{R}(B)$ if and only if there exists a point $\mathbf{y} \in B$ and an admissible tuple of parameters $\tilde{\phi} := (\tilde{\phi}_1, \ldots, \tilde{\phi}_n)^T \in A$ with

$$\mathbf{x} = \mathbf{t_y}(\tilde{\phi}) := \mathbf{R}(\tilde{\phi})\mathbf{y} + \mathbf{t}(\tilde{\phi}) \tag{2}$$

Equivalently we can define $\mathcal{R}(\mathbf{y})$ as the reachable space of a singular point $\mathbf{y} \in B$. Obviously $\mathcal{R}(\mathbf{y})$ is equivalent to the range of values $\mathbf{t_y}(A)$ over all admissible parameter tuples, so we have $\mathcal{R}(B) = \cup_{\mathbf{y}\in B}\mathbf{t_y}(A)$. The computation of the range of values[6] over an interval is NP-hard even for polynomial functions, see (Gaganov, 1985). As $\mathcal{R}(B)$ is a compact set and can be represented by its boundary, we are interested in computing the surface of $\mathcal{R}(B)$ and use the fact

$$\partial\mathcal{R}(B) \subseteq \bigcup_{\mathbf{y}\in\partial B} \mathbf{t_y}(A) \tag{3}$$

[5]By *efficiently* we mean, that not every face should be considered individually

[6]In general it is even not possible to compute it *exactly*, for it can be reduced to global optimization that is not solvable with finite convergence unless the algorithm uses the exact value of the Lipschitz constant.

Furthermore $\partial\mathcal{R}(B)$ cannot be solely constructed by combining $\partial\mathcal{R}(\mathbf{y})$ for a specific reference point $\mathbf{y} \in B$ with the geometric information of B, i.e. in general it is not possible to compute $\mathcal{R}(B)$ by only tracking the boundary of the workspace for a single reference point.

The reachable space $\mathcal{R}(\mathbf{y})$ of a singular point $\mathbf{y} \in B$, just as the reachable space of a body B, can be very complex in nature. In general it contains holes, i.e. it has a nonzero topological genus, and empty spaces that are not visible from outside called *voids*. The point set $\mathcal{R}(\mathbf{y})$ is path-connected, unless the mechanism has closed loops.

For the case of a singular point $\mathbf{y} \in B$, we can derive a criterion, whether $\mathbf{t}_\mathbf{y}(\tilde{\phi})$ lies on the surface of $\mathcal{R}(\mathbf{y})$ at a certain configuration $\tilde{\phi}$. We simplify the notation by considering the translational part of the forward kinematic of the reference point $\mathbf{t} := \mathbf{t}_\mathbf{0}$ of B. Unfortunately the criterion only works in one direction: If the criterion holds, $\mathbf{t}(\tilde{\phi})$ does not lie on the surface. As the criterion is based on a local argument, the other direction may be false as a result of global effects. In our context the inverse function theorem states, that there exists a small open subset around $\tilde{\phi} \in D$, such that $\mathbf{t}$ is a diffeomorphism and locally reversible, if its jacobian $\mathbf{J_t}(\tilde{\phi}) \in \mathbb{R}^{3\times n}$ has full rank. As a direct implication, we have for all[7] $\tilde{\phi} \notin \partial D$

$$\mathrm{rk}(\mathbf{J_t}(\tilde{\phi})) = \max! \;\Rightarrow\; \mathbf{t}(\tilde{\phi}) \notin \partial\mathbf{t}(D) \tag{4}$$

Considering the situation under the constraints $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m, \mathbf{c} \equiv \mathbf{0}$, we are able to apply the same arguments, resulting in

$$\mathrm{rk}\begin{pmatrix}\mathbf{J_t}(\tilde{\phi})\\ \mathbf{J_c}(\tilde{\phi})\end{pmatrix} = \max! \;\Rightarrow\; \mathbf{t}(\tilde{\phi}) \notin \partial\mathbf{t}(\{\phi \in D \mid \mathbf{c}(\phi) = \mathbf{0}\}) \tag{5}$$

2.3 Computational Tools

2.3.1 Interval Mathematics Interval mathematics analyses the extension of the concept of real numbers to real intervals whose set is denoted by $\mathbb{I}$. A good treatment of this subject can be found in (Moore, 1963). The methods presented in this paper are based on the estimation of the reachable space of a body B over "small" intervals of admissible parameters. As discussed above, the reachable space $\mathcal{R}(\mathbf{0})$ of the reference point of B over an interval $X = [\underline{X_1}, \overline{X_1}] \times \cdots \times [\underline{X_n}, \overline{X_n}] \in \mathbb{I}^n$ is identical to the range $\mathbf{t}(X)$ of its translational part of the forward kinematic. We get an *inclusion function* $I_\mathbf{t} : \mathbb{I}^n \longrightarrow \mathbb{I}^d$ which is characterized by the property $\mathbf{t}(X) \subseteq I_\mathbf{t}(X)\ \forall X \in \mathbb{I}^n$ by applying *interval arithmetic* on $\mathbf{t}$. Each component t_i can be described in the form of an arithmetic expression, composed of constants, elementary operations $\circ \in \{+,-,*\}$ and basic functions ω[8]. By extending

$$\omega(X) = \Box\{\omega(x) \mid x \in X\}, X \circ Y = \Box\{x \circ y \mid x \in X, y \in Y\}, \tag{6}$$

to operate on intervals we define the *natural interval extension* I_{t_i} for t_i and get $I_\mathbf{t} := (I_{t_1}, \ldots, I_{t_d})^T$. Here $\Box S$ denotes the smallest interval containing a set $S \subset \mathbb{R}^n$. The set of all intervals over S is denoted by $\mathbb{I}(S)$. The natural interval extension is *inclusion isotone*: $I_\mathbf{t}(Y) \subseteq I_\mathbf{t}(X)\ \forall Y \subseteq X$. The *width* of an interval quantifies its size in the maximum norm, $w(X) := \max_i\{\overline{X_i} - \underline{X_i}\}$. The function $\mathbf{t}$ is called *Lipschitz on S*, if there exists a positive constant L with

$$w(I_\mathbf{t}(X)) \le L \cdot w(X)\ \forall X \in \mathbb{I}(S).$$

A general way to get better inclusion functions is the consideration of derivatives. We call

$$M_\mathbf{t}(X) = f(C) + (X - C)^T I_{\mathbf{J_t}}(X) \tag{7}$$

the *mean value form* for $\mathbf{t}$, where C denotes the center of X. It has the inclusion property as can be seen by the mean value theorem of analysis. The *excess-width* $w(I_f(X)) - w(f(X))$ of the mean value form converges quadratically to zero with $w(X) \to 0$, whereas the convergence is linear in the case of the natural interval extension, which in turn can be faster evaluated.

The methods described below are based on the subdivision of intervals and the computation of estimations. Such recipes are also used by interval branch&bound methods for global optimization, e.g. see (Hansen, 1992). In particular, we constructively compute the range of a vector-valued function over $X \in \mathbb{I}^n$. No literature on exactly this problem could be found by the authors; for the real-valued case see (Asaithambi et al., 1982).

2.3.2 Bounding Object Hierarchies The most efficient way to handle great amounts of data is the use of hierarchical data structures, see (Samet, 1990). In our context *bounding object hierarchies* will prove themselves as very useful, as they already did in applications such as real-time collision detection, see (Hubbard, 1995). The reader can imagine these data structure as a representation of geometric data by simple enclosing objects with many levels of details. Their usage enables algorithms to focus on the relevant parts of a complex object, thus increasing the efficiency. We consider bounding object hierarchies as trees whose nodes represent bounding objects for subsets of a given geometric set. The sons of each inner node represent bounding objects together enclosing the same data as their father. The geometric primitives are usually stored in the leafs.

We distinguish between a *surface-* and a *volume-based* representation. In the first case, only the boundary representing the object is stored, whereas in the second case the entire volume of the object is considered. Later we discuss the generation of a surface-based hierarchy using spheres as bounding objects, it is called *sphere-tree*.

[7]if $\tilde{\phi} \in \partial D$ holds, $\mathbf{t}(\tilde{\phi})$ may yet be a surface point, caused by a joint limit.
[8]e.g. trigonometric functions or the absolute value

3 Pre-Computation

The computation of the reachable space $\mathcal{R}(B)$ starts with a pre-computational step. Similar to other methods, our main procedure demands an abstract mathematical input in the form of a set of functions. A very elegant way is to pre-compute all functions symbolically and to store them in *operator trees*. These data structures are evaluated recursively either in a standard way to get the function-value or with interval arithmetic. This realizes the desired inclusion functions on which our algorithms are based.

The information that geometrically describes B should be represented by its boundary (B-rep.) by a set of faces. For a *volume-based* computation, B has to be a solid and the set of faces must describe a polyhedron in a topologically correct way. This may cause difficulties, for the automated export of CAD data is still not fully solved, e.g. the tessellation process generates small gaps between adjacent faces (*cracks*) in cases where no topological information is stored with the CAD data. In this paper we focus on a *surface-based* computation which regards each face independently. Hence it is more tolerant regarding such effects. We hierarchically store the set of faces in a sphere-tree whose construction is described in subsection 4.3.2.

In summary we expect a mechanism to be given as a set of bodies and joints with the following information.

1. The geometry of each body is described by its boundary, by a set of faces.
2. For each kinematic link its technical data is known, including references to the pair of incident bodies.

The technical data of a joint consists of its type that also gives us the number of its free parameters, the joint range, and its transformation relative to the linked bodies. One of the given bodies has to be declared as the *base* that has a fixed position and orientation.

3.1 Determination of Forward Kinematics

Let $J_{i,j}$ denote the joint that connects body B_i and body B_j. This structure can be modeled as an undirected graph by identifying each part with a node and connecting a pair of nodes if the corresponding joint exists. The node that is assigned to the base body is called *source*. If this graph contains any cycles, the first step is to compute its spanning tree. Once we have a tree structure, there exists a unique path from the source to each node that enables us to uniquely assign a forward kinematic to each body.

Let $\mathbf{A}_{i,j} \in \mathbb{R}^{4\times4}$ be the homogeneous matrix describing the relative transformation from the body reference frame of B_i into the joint frame of $J_{i,j}$[9] and let $\mathbf{T}_{i,j} : \mathbb{R}^k \longrightarrow \mathbb{R}^{4\times4}$ describe the characteristic transformation of $J_{i,j}$. This transformation and its number of variables is solely defined by the joint type. With this notations the forward kinematic of B_i is given by

$$\mathbf{f}_i(\phi_1,\ldots,\phi_{l+k}) = \mathbf{f}_{i-1}(\phi_1,\ldots,\phi_l)\cdot\mathbf{A}_{i-1,i}\cdot\mathbf{T}_{i,i-1}(\phi_{l+1},\ldots,\phi_{l+k})\cdot\mathbf{A}_{i,i-1}^{-1}, \qquad (8)$$

where $\mathbf{f}_{i-1}$ represents the forward kinematic of B_{i-1}, the unique predecessor of B_i in the spanning tree. These dependencies are determined by applying breadth-first search starting at the source node.

Each evaluation of $\mathbf{f}$ necessitates a re-calculation of (8) with constant arguments. We avoid this for the reason of efficiency and perform this computation symbolically. The idea is realized by storing each single operation in an operator tree. Each inner node of the tree symbolizes an arithmetic operation or a primitive function whose number of arguments defines the out-degree of the node. Each leave of the tree stores a constant or a variable. The different priorities of the operations are hidden in the hierarchy of the tree. The way of evaluation is independent of the operator tree itself and can be regular or e.g. done by using interval arithmetic realizing a natural inclusion function for the corresponding function. Usually the tree is simplified before its first use by e.g. deleting neutral operations or summing up operations on constants.

3.2 Determination of Constraints

The abstract representation of the kinematic behavior of each body of the mechanical system is still incomplete if its kinematic graph contains cycles. In this case there exist *non-tree edges*, edges contained in the graph but not in its spanning tree. Let $J_{i,j}$ represent such an edge, then $\mathbf{f}'_i = \mathbf{f}_j\cdot\mathbf{A}_{j,i}\cdot\mathbf{T}_{i,j}\cdot\mathbf{A}_{i,j}^{-1}$ is an alternative valid forward kinematic for body B_i. In order to keep the loop closed at $J_{i,j}$, we have to equate both forward kinematics, thus getting the constraint[10]

$$\mathbf{c} := \mathbf{f}_i - \mathbf{f}'_i \equiv \mathbf{0} \qquad (9)$$

We calculate (9) for each non-tree edge and obtain a set C of all constraints. These have to be satisfied in order to keep all the loops closed in the system.

The main computation discussed later considers each body B separately. In general it is not the case that B is restricted by every loop. We can save many computations, if we neglect those constraints whose violation has no effect on B. In order to determine a minimal set of constraints for B, we observe that C may be subdivided into subsets that are independent of each other in the same way as e.g. $f(x,y) = 0$ and $g(z) = 0$ can be investigated and satisfied independently. This subdivision of C into $C_1,\ldots,C_k$ can be determined by collecting constraints according to the indices of their parameters such that no constraint is

[9]the transformation from B_j to $J_{i,j}$ is denoted by $\mathbf{A}_{j,i}$

[10]By calculating with quaternions we get seven individual real-valued constraints.

contained in two subsets. In a second step, for each body B we compare the indices of its forward kinematic $\mathbf{f}(\phi_1, \ldots, \phi_n)$ with those of $C_1 \ldots, C_k$. For $i = 1, \ldots, k$ the subset C_i of constraints restricts the position and orientation of B if one of the variables $\phi_1, \ldots, \phi_n$ also appears in C_i. This can be tested by a comparison of the indices analogously. The set operations on indices can efficiently be implemented by using a *union-find* data-structure, see (Mehlhorn, 1984). For each body B, we store the minimal set of constraints $\mathbf{c} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$.

4 Hierarchical Approximation

The idea of *hierarchical approximation* is the estimation of the forward kinematic of a body B over (hyper-)intervals generated by a hierarchical subdivision of the parameter space. If the mechanism has closed loops, the pre-computed constraints are checked simultaneously. Both tasks are realized with the help of inclusion functions. The result is an approximation of the reachable space of the reference point of B. We use the pre-computed bounding object hierarchies to extend our methods to take the geometric description of B into account.

The *volume-based* variant of our method is capable of computing an enclosing ε-approximation of the reachable space, where ε denotes a tolerance pre-defined by the packaging engineer. However we will emphasize on the *surface-based* variant which computes an envelope for $\mathcal{R}(B)$. This is more efficient and nevertheless exactly meets the demands of practical applications. The acceleration is achieved by the use of surface-based bounding object hierarchies and a kind of *lazy evaluation*: we focus on intervals whose estimation contribute to the surface of the present approximation[11].

For a clear understanding of the basic principles, we first explain the computation of the reachable space of the reference point of a body that is not associated with constraints. We then show that this method can be extended to obey closed-loop constraints and to work with regard to complex geometric shapes.

4.1 The Case of Tree-Structures

Let $\mathbf{t} : \mathbb{R}^n \longrightarrow \mathbb{R}^3$ be the translational part of the forward kinematic for some body B that is part of a tree-structured mechanism. As already explained in subsection 2.3.1, the reachable space of the reference point of B is exactly given by the range of values $\mathbf{t}(D)$. In the case of a tree structure, $D \in \mathbb{I}^n$ represents the space of all admissible parameters. In order to approximate $\mathbf{t}(D)$ we subdivide D into a set $\mathcal{L}$ of intervals $X_1, \ldots, X_l$ with $D = \cup_{X \in \mathcal{L}} X$ and estimate the range of values of $\mathbf{t}$ over X_i by computing $I_\mathbf{t}(X_i)$. The inclusion property and inclusion-isotony of $I_\mathbf{t}$ imply $\mathbf{t}(D) \subseteq \cup_{X \in \mathcal{L}} I_\mathbf{t}(X) \subseteq I_\mathbf{t}(D)$. This procedure is sketched in figure 1.

[11]This may lead to the overlook of voids that are contained inside $\mathcal{R}(B)$. Thus it is not suited for the volume-based case.

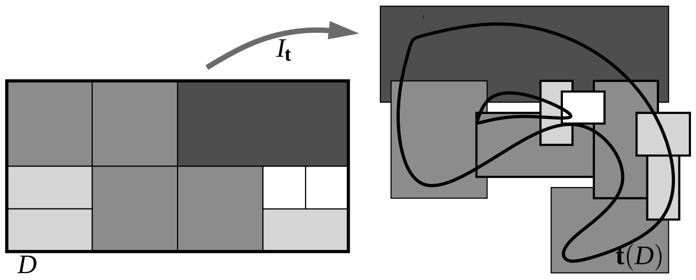


Figure 1. The parameter space D is subdivided (left) and estimated by an inclusion function $I_\mathbf{t}$ yielding an approximation of $\mathbf{t}(D)$ (right). Various shadings symbolize different generations of intervals.

Obviously $\mathbf{t}$ is Lipschitz, so we also have a constant L with $w(I_\mathbf{t}(X)) \leq L \cdot w(X)\ \forall X \in \mathbb{I}(D)$. Hence for any $\delta > 0$ there exists a subdivision $\mathcal{L}$ of D such that $w(I_\mathbf{t}(X)) \leq \delta\ \forall X \in \mathcal{L}$ holds. Then the minimal distance between some point of our approximation $\cup_{X \in \mathcal{L}} I_\mathbf{t}(X)$ and $\mathbf{t}(D)$ is at most $\sqrt{3}\delta$. Thus we have an ε-approximation[12] if we set $\varepsilon := \sqrt{3}\delta$. We can also say that the approximation converges to $\mathbf{t}(D)$ with $w(X) \longrightarrow 0\ \forall X \in \mathcal{L}$.

As already mentioned an important accelerating factor is the disregard of intervals whose estimation does not contribute to the outer surface of the present approximation. In other words: we subdivide an interval $X \in \mathcal{L}$ only if one of the faces of $I_\mathbf{t}(X)$ is part of the outer surface of the present approximation. Observe that an interval once neglected may contribute to the outer surface of a refined approximation. We get the following termination criterion for our algorithm

$$\forall X \in \mathcal{L} : I_\mathbf{t}(X) \cap \partial_o(\cup_{X \in \mathcal{L}} I_\mathbf{t}(X)) \neq \emptyset : \ w(I_\mathbf{t}(X)) \leq \varepsilon/\sqrt{3} \quad (10)$$

with ∂_o as the denotation for the *outer surface* of a compact set[13].

Above we use the addendum ”outer”, because the range of any interval X can contain voids bounded by surfaces that may not be found unless X is subdivided further. Thus we are not guaranteed to find any ”inner” surfaces when using lazy evaluation, so we restrict ourselves to the ”outer” surface. To be more precise, what we compute is the outer hull (envelope) of an ε-approximation[14]. See the left picture in figure 2.

Since $\mathbf{t}(D)$ is a compact set and can be represented by its boundary, we may in addition exclude any interval X which produces no boundary points, i.e. $\mathbf{t}(X) \cap \partial \mathbf{t}(D) = \emptyset$. We check this by using observation (4) in conjunction with interval analysis. In the same way we get inclusion functions for the forward kinematics and for constraints, we pre-compute inclusion functions for the determinants of any maximal submatrix of the jacobian

[12]This is equivalent to $\delta_H(\cup_{X \in \mathcal{L}} I_\mathbf{t}(X), \mathbf{t}(D)) \leq \varepsilon$ where δ_H denotes the *Hausdorff distance*.
[13]We could also use the diameter in (10): $\mathrm{diam}(I(X)) \leq \varepsilon$.
[14]This should not to be mixed up with the ε-approximation of the outer surface.

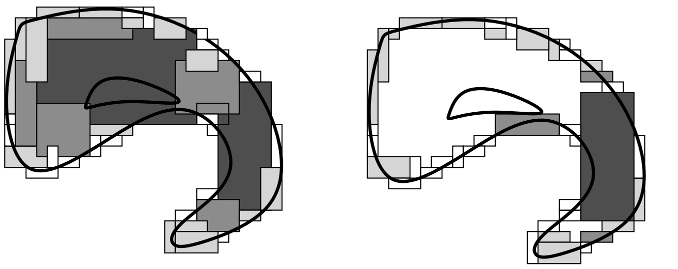


Figure 2. The effect of lazy evaluation (left) and the additional omission of intervals that produce no boundary points (right).

of $\mathbf{t}$. This effort is worthwhile only for inputs with a moderate number of parameters $n \geq 3$. In this case the number of 3×3 submatrices of the jacobian $\mathbf{J_t}$ is $\binom{n}{3}$.

Let M_i be the i-th of these matrices and $I_{|M_i|}$ an inclusion function for its determinant. In order to exclude an interval X with $X \cap \partial D = \emptyset$, we test these functions with the argument X. As soon as we have $\mathbf{0} \notin I_{|M_i|}(X)$ for one of these functions, we know that the corresponding submatrix is non-singular for all parameters $\phi \in X$. So X does not produce any boundary points. It holds for $X \cap \partial D = \emptyset$

$$\begin{aligned} &\exists M_i : \mathbf{0} \notin I_{|M_i|}(X) \\ \Rightarrow\ & I_{\mathbf{t}}(X) \cap \partial \mathbf{t}(D) \neq \emptyset \end{aligned} \tag{11}$$

If X lies on the surface of D, the situation is slightly more difficult. This is described in (Fuhrmann, 2001). The result of this effort is illustrated in figure 2.

Algorithm EnvelopeOfReachableSpace $(\varepsilon, D, I_{\mathbf{t}})$
Input. Tolerance $\varepsilon > 0$, starting interval $D \in \mathbb{I}^n$, interval function $I_{\mathbf{t}}$
Output. List $\mathcal{L}$

```
1   $X := D$; $\mathcal{L} := \mathcal{L}_o := X$;
2   do
*
3       for some $X \in \mathcal{L}_o$ with $w(I_{\mathbf{t}}(X)) > \varepsilon/\sqrt{3}$
4           delete $X$ from $\mathcal{L}$;
5           bisect $X$ into $X_1$ and $X_2$ parallel to its longest edge;
6           for $i = 1, 2$ do
7               if $X_i$ may produce surface points, i.e. (11) holds
                then insert $X_i$ into $\mathcal{L}$;
8       update $\mathcal{L}_o := \{X \in \mathcal{L} \mid I_{\mathbf{t}}(X) \cap \partial_o(\cup_{X \in \mathcal{L}} I_{\mathbf{t}}(X)) \neq \emptyset\}$;
9   until termination criterion (10) holds;
10  return $\mathcal{L}_o$;
```

Algorithm 1

We conclude: *The outer surface of $\cup_{X \in \mathcal{L}_o} X$ computed by algorithm 4.1 is the outer surface of an ε-approximation to $\mathbf{t}(D)$.* The final approximation consists of an estimation of all relevant intervals. The boxes on its outer boundary are small enough, i.e. (10) holds. We precisely investigate the correctness and running time of the algorithm in (Fuhrmann, 2001).

The computation of the surface of a set of boxes, also called their *contour*, is described in (Preparata and Shamos, 1985) for the two-dimensional case. To realize step 8, it is not necessary to compute the entire surface each time. We use the following idea in our implementation. The present set of boxes is maintained in a hierarchical data-structure in the form of coordinates, e.g. an oct-tree or segment-trees. If a box b has to be deleted, we delete it from this data-structure and consider all boxes that intersect with b as new boxes. For a new box b, we have to determine whether it contributes to the present surface. The data-structure is used to find all boxes having a non-empty intersection with b. All these boxes "cut away" a portion of the surface of b. Obviously, b contributes to the surface and belongs to $\mathcal{L}_o$, if there is a portion of ∂b that does not lie in any other box.

This procedure does not distinguish between inner and outer surface[15], but it is efficient regarding possible alternatives and does not affect the correctness of the algorithm. Figure 3 shows the result of our implementation applied to open kinematic chains.

4.2 How to Handle Constraints

If the mechanism has closed kinematic loops, we precompute a set of constraints $\mathbf{c} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ as described in subsection 3.2. These constraints define the set of admissible parameters $A := \{\phi \in D \in \mathbb{I}^n \mid \mathbf{c}(\phi) = \mathbf{0}\}$. For $\mathbf{c}$ is highly nonlinear in general, it is a very hard task to find some elements of A, nothing to say of exactly determining A entirely. The idea is to approximate A by using interval methods as above. For a better

[15] $\mathcal{L}_o$ may also contain intervals X for which $I_{\mathbf{t}}(X)$ contributes to the surface of a void.

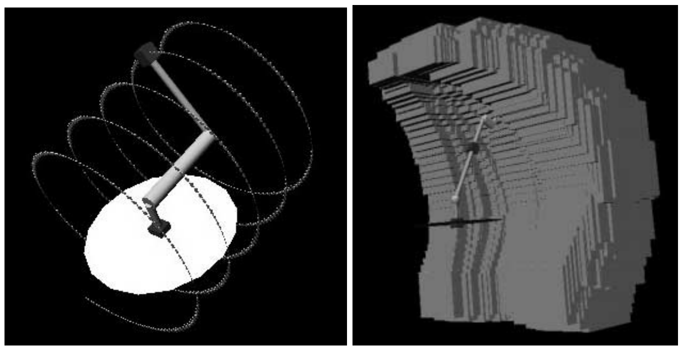


Figure 3. Left: A screw joint serves as a simple example to clarify the correctness of the method. Right: An example with 4dof.

understanding we first explain this as an independent procedure which may take place as a pre-computational step. The result is a set $\mathcal{A} \subset \mathbb{I}(D)$ of intervals with the following properties.

(i) All admissible points are enclosed: $A \subseteq \cup_{X \in \mathcal{A}} X$
(ii) No constraint is violated more than $\varepsilon_{\mathbf{c}}$:

$$\forall \phi \in X \in \mathcal{A} : -\varepsilon_{\mathbf{c}} \leq c_i(\phi) \leq \varepsilon_{\mathbf{c}} \; \forall i = 0, \ldots, m-1$$

The set $\mathcal{A}$ is computed by generating an inclusion function $I_{\mathbf{c}}$ which can be used to test an interval for being not admissible. We have

$$\mathbf{0} \notin I_{\mathbf{c}}(X) \;\Rightarrow\; \nexists \phi \in X : \mathbf{c}(\phi) = \mathbf{0}$$

However $\mathbf{0} \in I_{\mathbf{c}}(X)$ does not imply that X contains any admissible point. In order to compute $\mathcal{A}$, we successively subdivide D into a list $\mathcal{L}$ of intervals as above and exclude all intervals $X \in \mathcal{L}$: $\mathbf{0} \notin I_{\mathbf{c}}(X)$. We satisfy specification (ii) by using the termination criterion $\forall X \in \mathcal{L} : w(I_{\mathbf{c}}(X)) \leq \varepsilon_{\mathbf{c}}$.

There is no reason why this procedure and algorithm 4.1 should not work simultaneously. This is even more efficient because any interval that is neglected by lazy evaluation also does not have to be tested for validity further. The observing reader might already have noticed that in general forward kinematics and constraints neither depend on all parameters of the system nor do they exactly depend on the same parameters. Let F denote the (high-dimensional) interval that lies in D and represents the parameter space of the variables that actually occur in the expression $\mathbf{f}$. In the same way let C be the interval defined by joint limits for the variables in $\mathbf{c}$. Both intervals have a common sub-interval $S := F \cap C$, for $\mathbf{f}$ and $\mathbf{c}$ must have some parameters in common.

We now consider algorithm 4.1 as working on F instead of D and we call an interval $X \in \mathbb{I}(F)$ *temporarily admissible* if

$$\exists Y \in \mathbb{I}(C) : \mathrm{proj}_S(Y) \subseteq \mathrm{proj}_S(X) \tag{12}$$

where $\mathrm{proj}_S(X)$ denotes the projection of an interval X onto the subspace S. We call X $\varepsilon_{\mathbf{c}}$*-admissible* if (12) holds together with $w(I_{\mathbf{c}}(Y)) \leq \varepsilon_{\mathbf{c}}$. In practice the kinematic loops have to be closed very accurately, so we may expect the relation $\varepsilon_{\mathbf{c}} < \varepsilon$. This means we have to subdivide C and F with different granularities. We extend algorithm 4.1 at the place "*" by the following procedure. It maintains a list $\mathcal{L}_C$ that represents a sufficiently fine subdivision of C excluding irrelevant intervals. All intervals $X \in \mathcal{L}_o$ being not temporarily admissible are removed. Together with an extension of termination criterion (10), this ensures that C is subdivided faster than F, resulting in $\varepsilon_{\mathbf{c}}$-admissible intervals where they are needed. We use

$$\forall X \in \mathcal{L} : I_{\mathbf{t}}(X) \cap \partial_o(\cup_{X \in \mathcal{L}} I_{\mathbf{t}}(X)) : \\ w(I_{\mathbf{t}}(X)) \leq \varepsilon \wedge X \text{ is } \varepsilon_{\mathbf{c}}\text{-admissible} \tag{13}$$

Algorithm *

```
1  do
2      choose $Y \in \mathcal{L}_C : \exists X \in \mathcal{L}_o : \mathrm{proj}_S(X) \subseteq \mathrm{proj}_S(Y)$ with
       maximal width and delete $Y$ from $\mathcal{L}_C$;
3      bisect $Y$ into $Y_1$ and $Y_2$ parallel to its longest edge;
4      for $i = 1,2$ do
5          if $\mathbf{0} \in I_{\mathbf{c}}(Y_i)$ then insert $Y_i$ into $\mathcal{L}_C$;
6  until $\nexists X \in \mathcal{L}_o : \exists Y \in \mathcal{L}_C : \mathrm{proj}_S(X) \subseteq \mathrm{proj}_S(Y)$;
7  forall $X \in \mathcal{L}_o$ do
8      if $X$ not temporarily admissible then delete $X$ from $\mathcal{L}, \mathcal{L}_o$;
```

Algorithm 2

Here, the deletion of intervals that do not produce boundary points in line 7 of algorithm 4.1 can be retained according to the same principle (11). As we see by (5) we now have to consider submatrices of a higher dimension. This results in more complex formulae and the inclusion function may be very over-estimating until the intervals are very small. Hence we use a more sophisticated method for analyzing the minors, described in (Fuhrmann, 2001). We implemented the above algorithm and applied it successfully to different problems as can be seen in figure 4.

4.3 How to Handle Complex Geometries

So far, the above method is very general referring to the kinematic structure, but it only computes the reachable space for a singular point of a part. The next step is the extension towards sets of triangles that are hierarchically represented by a pre-computed sphere-tree. For a better understanding we postpone the usage of the hierarchy and first explain the basic case.

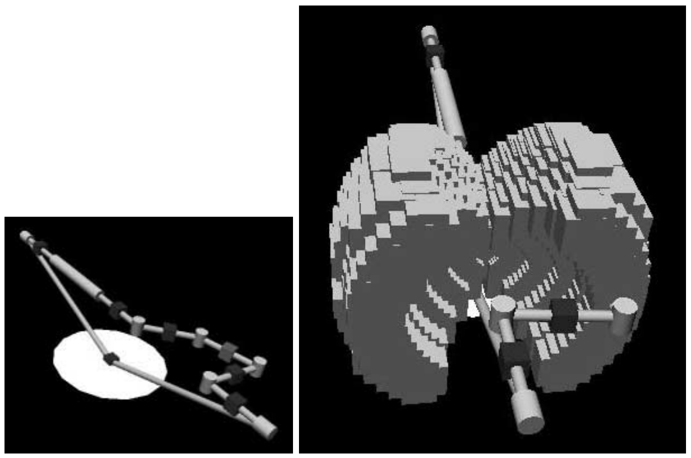


Figure 4. The reachable space of the reference point of a part of a 7dof closed-loop mechanism (simple model of a steering column).

4.3.1 The Primitives

Let the forward kinematic of a body B be given by (1) and let one of its triangles T be given by a point $\mathbf{p} \in \mathbb{R}^3$ and two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$. We then are able to derive this equation[16] for the forward kinematic of each point of T

$$\mathbf{t}_T(\alpha,\beta,\phi) := \mathbf{R}(\phi)\cdot(\mathbf{p}+\alpha\mathbf{a}+(1-\alpha)\beta\mathbf{b})+\mathbf{t}(\phi) \qquad (14)$$

with $\alpha, \beta \in [0;1]$. By this marginal extension of our present method and its application to each triangle of B, we could already compute the reachable space $\mathcal{R}(B)$. With increasing number of triangles the support of the computation by the pre-computed sphere-tree becomes advantageous.

4.3.2 Building and Using a Sphere-Tree

Assuming the geometry of each body B is described by its surface in the form of a set of triangles $\mathcal{T}$ for which we pre-computed a bounding object hierarchy based on spheres. The idea of applying a sphere tree is to compute the reachable space of a coarse approximation of B by spheres. This approximation is refined, i.e. the hierarchy is stepped down, only at spheres whose estimation contributes to the surface. In this way we find the relevant subsets of $\mathcal{T}$ with regard to a considered interval of parameters. Thus the usage of the bounding object hierarchy is again based on lazy evaluation[17]. Spheres have the advantage of rotational invariance, such that only the translational part of the forward kinematic has to be estimated.

We first sketch the recursive top-down generation of a binary sphere-tree for $\mathcal{T}$. Let the node S of the hierarchy be represented by a sphere which encloses $\mathcal{T}' \subseteq \mathcal{T}$. We cut $\mathcal{T}'$ into two disjunct subsets $\mathcal{T}_1' \cup \mathcal{T}_2' = \mathcal{T}'$ of almost equal complexity in a reasonable manner and generate two sons S_1, S_2 of S. For $i = 1,2$ the node S_i is represented by the smallest enclosing sphere for the vertices of $\mathcal{T}_i'$; we propose the usage of the algorithm of (Welzl, 1991) for this computation. If there is only one triangle T in $\mathcal{T}_i'$ left, the node S_i is called *leaf* and is represented by T.

Usually each sphere is stored by its radius and its center point $\mathbf{p} \in \mathbb{R}^3$ relative to the respective body reference frame. In our context we have to pre-compute the translational part of the forward kinematic of each center point which is given by

$$\mathbf{t}_\mathbf{p} : \mathbb{R}^n \longrightarrow \mathbb{R}^3; \mathbf{t}_\mathbf{p} := \mathbf{R}(\phi)\mathbf{p}+\mathbf{t}(\phi)$$

using the notations of (1). Thus the geometry is represented by a binary tree in which each node stores a forward kinematic and a radius. Each leaf additionally contains a triangle.

[16]The triangle T is given by $\mathbf{p}+\alpha\mathbf{a}+\beta\mathbf{b}$ with $\alpha,\beta \in [0;1]$ and $\alpha+\beta \leq 1$. So the (α,β)-space is the basic 2-simplex and we have to re-parameterize using

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} := \begin{pmatrix}\alpha' \\ (1-\alpha')\beta'\end{pmatrix} \text{ with } \alpha',\beta' \in [0;1]$$

[17]We see no way to use the hierarchy in a more classical fashion, e.g. a disregard of complete subtrees controlled by cut-off tests.

Next we explain how to estimate the reachable space of a sphere S given by its radius r and its forward kinematic $\mathbf{t}_\mathbf{p}$. Given an interval $X \in \mathbb{I}(F)$ of admissible parameters, we compute the box $I_{\mathbf{t}_\mathbf{p}}(X) \subset \mathbb{R}^3$ and move its faces outwards by r. We denote the result by $E(S,X)$. Observe that it contains the reachable space over X of all triangles enclosed by S.

Now assume we want to estimate the reachable space of B over the fixed interval X by using the sphere-tree. This may be realized by applying the ideas already used in 4.1. We maintain a list of estimations, and in each iteration we choose and delete the element $E(S,X)$ with the maximal width of all estimations that contribute to the outer surface of the present approximation. If S has two sons S_1 and S_2, we compute $E(S_1,X)$ and $E(S_2,X)$ and insert them into the list. If S_i represents a leaf of the sphere-tree, we compute the estimation $E(T,X)$ given by (14) for the triangle T stored in this leaf. The iteration terminates if all estimations that lie on the surface are generated from triangles. This gives us a good impression on how the idea of lazy evaluation is used in the context of sphere-trees to increase efficiency.

The outer surface of $\mathcal{R}(B)$ is approximated by combining this idea with algorithm 4.1. Instead of considering a fixed interval, the idea is to perform some iterations of algorithm 4.1 to subdivide F and C and to test the given constraints. The resulting subintervals are then estimated in the manner described above and the process is iterated.

So, abstractly speaking the over-all algorithm has to pass through three hierarchies[18] simultaneously. It is not clear through theoretical considerations how to assess the respective hierarchy, i.e. how fast it should be traversed. This has to be evaluated experimentally by a performance analysis. Intuitively the best way seems to be a uniform traversal, such that the leafs of the sphere-tree are reached at the same time as the granularity of the subdivisions guarantees the given tolerances.

4.3.3 Self-Collisions

A further interesting topic is the computation of the reachable space sensitive to self-collisions of the mechanism[19]. We could use our methodology to pre-compute an approximation to the set of parameters that produce self-collisions, but unfortunately this is very time-consuming. To this purpose we would investigate all pairs of bodies B, B', their corresponding sphere-trees and the union of the parameter spaces of their forward kinematics. Regarding two estimations $E(S,X), E(S',X)$, the idea is to subdivide X and to descend in the trees at S and S' only if the estimated intervals intersect.

[18]These are the parameter spaces F and C and the sphere tree

[19]in the sense that the reachable space is computed only considering non-colliding configurations.

5 Conclusion and Future Work

We have presented a method for computing a ”minimal” envelope that contains a given mechanism in all its admissible configurations. To be precise, the envelope is the outer surface of an ε-approximation of the reachable space of the mechanism as defined in subsection 2.2. The method is very general: it handles all mechanisms that can be defined in the sense of multibody systems; the geometry of a part is expected to be representable as a set of triangles. The approximation is conservative and globally correct, i.e. it always contains the reachable space and any desired accuracy can be achieved, only limited by computation-time and -space. Our present implementation shows to us, that the method can be implemented to work very robustly. We can also use our approach to approximate the swept volume of a single object with any granularity: Its trajectory has to be given as a parameterized curve and then is treated as a special joint (a guide) subject to which the object is movable.

The critical point of our method is surely its efficiency, even if we already try to exploit several algorithmic and mathematical properties of the problem. Hence our future work will mainly concentrate on improving the running-time of the algorithm. We enumerate some topics that have to be considered and experimentally evaluated by an implementation.

1. There are inclusion functions with a smaller excess-width, e.g. the mean value form (7). However their evaluation often is more time-consuming, so a clever trade-off has to be found.
2. The simultaneous computation for many bodies could be faster than considering each body individually, for there are many indications that the running-time of our algorithm strongly depends on the size of the approximated surface. Which number of bodies that maximizes the efficiency?
3. Certain closed-loop systems commonly occur as submechanisms, e.g. four-bar mechanisms. For such cases we suggest the use of pre-computed forward kinematics that redundantize additional constraints.

6 Acknowledgements

The authors would like to thank Prof. Dr. Günter Hotz, head of the Insitute for Applied Mathematics and Computer Science, Universität des Saarlandes, Saarbrücken, for his encouragement and support on this ongoing research.

REFERENCES

Abdel-Malek, K., Adkins, F., Yeh, H. J., and Haug, E. (1997). On the determination of boundaries to manipulator workspaces. *Robotics and Computer-Integrated Manufacturing*, 13:63–72.

Abdel-Malek, K., Blackmore, D., and Joy, K. (2000). Swept volumes: Foundations, perspectives and applications. submitted to International Journal of Shape Modeling.

Abdel-Malek, K., Yeh, H. J., and Keirallah, N. (1999). Workspace, void and volume determination of the general 5dof manipulator. *Mechanics of Structures and Machines*, 27:91–117.

Asaithambi, N. S., Zuhe, S., and Moore, R. E. (1982). On computing the range of values. *Computing*, 28(3):225–237.

Blackmore, D., Leu, M., and Wang, W. (1992). Classification and analysis of robot swept volumes. In *Proc. Japan USA Symposion on Flexible Automation, Vol. 1*, pages 69–75.

Fuhrmann, A. (2001). Methoden zur Berechnung des erreichbaren Raumes beliebiger Mechaniken. Technical report, DaimlerChrysler Forschung und Technologie, Ulm, Germany.

Fuhrmann, A., Hotz, G., Sauer, J., and Schömer, E. (2000). Methodik zur Berechnung des maximal benötigten Bauraumes beliebiger Kinematiken. Patent description submitted to Intellectual Property Management of DaimlerChrysler AG.

Gaganov, A. A. (1985). Computational complexity of the range of the polynomial in several variables. *Cybernetics*, pages 418–421.

Gupta, K. C. (1997). *Mechanics and Control of Robots*. Mechanical Engineering Series. Springer, Berlin.

Hansen, E. (1992). *Global Optimization Using Interval Analysis*. Dekker, New York.

Haug, E. J. (1995). Numerical algorithms for mapping bounderies of manipulator workspaces. *Advances in Design Automation*, 69(2):447–459.

Hubbard, P. M. (1995). Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230.

Kumar, A. and Waldron, K. J. (1981). The workspaces of a mechanical manipulator. *Journal of Mechanical Design*, 103:665–672.

Mehlhorn, K. (1984). *Data Structures and Efficient Algorithms*. EATCS Monographs. Springer, Heidelberg.

Moore, R. (1963). *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

Preparata, F. P. and Shamos, M. I. (1985). *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY.

Rastegar, J. and Deravi, P. (1987). Methods to determine workspace, its subspaces with different numbers of configurations and all the possible configurations of a manipulator. *Journal of Mech. Mach. Theory*, 22(4):343–350.

Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*. Addison Wesley, Boston.

Wang, J. Y. and Wu, J. K. (1993). Dextrous workspaces of manipulators II: Computational methods. *Mechanics of Structures and Machines*, 21(4):471–506.

Wang, W. P. and Wang, K. K. (1986). Real-time verification of multi-axis nc programs with raster graphics. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 166–171.

Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). *Lecture Notes in Computer Science*, 555:359pp.