N90-29074

Organising Geometric Computations for Space Telerobotics

Stephen Cameron

Robotics Group, Department of Engineering Science
University of Oxford, Oxford OX1 3PJ, U.K.
E-mail: stephen@uk.ac.oxford.robots

Abstract

A truly intelligent system that interacts with the physical world must be endowed with the ability the compute with shapes: despite this spatial reasoning is rarely regarded as part of mainstream A.I. We argue that the study of "intelligent" spatial algorithms is a worthwhile activity, and give opinions and suggestions for the way forward.

1 What is Spatial Reasoning?

A truly intelligent system that interacts with the physical world must be endowed with the ability the compute with shapes. Despite this the study of "spatial reasoning" is a young field and is often confused with CAD/CAM. One problem is convincing people that the problem *is* hard: we liken it to vision, which is a problem that humans deal with easily, but has proved immensely difficult to solve on the computer. To see why spatial reasoning is of interest in the field of space telerobotics, consider the following scenarios:

1. An unmanned orbital maintenance robot is sent to replace a failed module in a communications satellite. The replacement is effected by a teleoperated manipulator, which is commanded to secure itself to the satellite, open an access door, bring out the old module, and replace it with the new one. Such teleoperations are commonplace in earth-bound activities which require the handling of hazardous materials; in the terrestrial case a human provides all the control of the manipulator via a wired teleoperator system. However, although it is technically feasible to provide the control channels to and from a human operator to an orbital vehicle, there is an unavoidable complication: the time lag introduced by the sluggishness of electromagnetic waves as a communication medium. A distance of only 15 000km introduces a delay of 0.1s, which is sufficient to make the manipulation task much harder. What is required here is the ability for the manipulator itself to deal with closing the low-level control loop: the manipulator take over the low-level actions such as "grip", "apply torque", and "open door". To effect these operations reliably requires a system that can reason over the geometric properties of its environment.

2. A Martian Rover vehicle is exploring a small part of the surface of the red planet. Although in constant communication with Earth it too suffers from the commands being sent from Earth being decided on the basis of information it sent some time ago—in this case, upward of half an hour. Even for such a vehicle to survive under such conditions it must at least be able to react to such immediate hazards as boulders or crevices in its path; to operate

efficiently it should be able to take some 'sensible' action in such cases (e.g., navigate around boulders). It is important to realise that the problem is not just one of recognising the hazard: once the hazard has been identified the appropriate corrective action depends crucially on its *geometry.*

3. A future space station is constructed by a number of astronauts with the help of a number of assembly robots. These robots are smarter versions of the maintenance vehicle (scenario 1) which are under the command of the astronauts, who use spoken commands made up from a small vocabulary to ease the data input problem. Thus these robots must be able to decide on how to parse crude assembly plans (as given by the astronauts) and produce detailed plans that they can execute. The parsing of these plans should take into account the relevance of the potential detailed plan to the overall mission (i.e., they should reject nonsensical parses) and, for reasons of safety, the robots should also try to forsee potential dangers to either the astronauts or the fabric of the space station. (Such a robot would probably require a good sense of dynamics and statics, as well as geometry.)

One thing that is clear from these examples is that all the tasks are "easy" for humans; we do them without conscious effort. However even the simplest of the tasks (scenario 1) would stretch our knowledge of how to write programs to deal with these problems. We first became interested in spatial problems through work in assembly robotics (e.g., [26,1]); however the problems occur in many guises. This paper explores the spectrum of such problems, and explains our first attempts to build systems that are truly spatially aware.

1.1 Types of Spatial Reasoning Problems

We can get a better grasp of the extent of spatial reasoning problems by listing some of the specific problems that have been tackled.

- *Storing the shape of objects.* This is an obvious, though non-trivial, prerequisite for most spatial reasoning problems. Storing shapes has been the subject of much research in the solid modelling community, though there the application is often to render pictures of objects.
- *Visual Recognition of Objects.* This problem has been studied for many years now, with limited, but steady, progress.
- *Interference Detection.* Given a number of objects in known positions and orientations, do they overlap?
- *Collision Detection.* Given a number of objects with know motions, do they collide?
- *Collision Avoidance.* Given a number of objects with start and goal positions, plan collision free motions.
- *Grasp Planning.* A specialised form of collision avoidance, where the goal is to obtain a stable grasp position.
- *Cloth Cutting.* Given a number of two-dimensional templates and a quantity of stock material, cut the items from the stock so as to minimise wastage.
- *Bin Packing.* Three dimensional analogue of cloth cutting, where we want to fit some parts into a bin. However if, say, the parts are electronic components which are to be connected

to form a device we would have further constraints on the solution, such as: minimising the wiring, allowing some parts to dissipate heat, and fixing controls near to the surface of the bin.

- *Pipe Routing.* Given an aircraft, plan routes for the various electrical ducts, hydraulic pipes, etc., while taking into account safety factors and other constraints.
- *Designing a Mechanical Assembly.* Here the input is a specification of the *function* of the assembly; the output should be a physical description of a suitable assembly (and preferably instructions as to how to make it).

It can be seen that spatial reasoning problems occur in a number of guises. The first problem (storing shapes) has been the subject of much research in the geometric modelling community, although it has often been with the main purpose of producing pictures of them. Some of the problems have reasonbly well understood solutions; collision detection has been studied in robotics, vision systems are in practical use, and grasp planning is possible for some manipulators. Some of the problems have been well studied in theory, though the solutions so found are still to be incorporated in a practical system (collision avoidance), while others are only solved by humans with the help of a computer (cloth cutting, bin packing, pipe routing).

2 Problems with Spatial Reasoning

It should be clear that there are a large number of practical problems that require spatial reasoning for their solutions; we were interested in looking for patterns that would help to find general solutions. Examination of the spatial reasoning problems—some on paper, some using computers—have lead us to identify certain generic problems and trends.

- *No single method of solution.* When solutions are known to these problems several different methods can be identified, with no one method being general. (For example, [9] describes three different ways of performing collision detection, and [12] describes another.)
- *Different Representations.* There is no single obvious way to store a shape, and different methods of solving spatial reasoning problems may require different representations, and thus have different functionality[1]. This lack of a normal form for shape descriptions means that it is difficult to write algorithms that work in all cases (as Murphy's Law ensures that whichever shape representation system you use, somebody else will want to use a different one), and conversion between the existing shape descriptions is slow at best and an unsolved problem at worst.
- *Computer Arithmetic is of Finite Precision.* Although we can compute quantities to whatever precision we like, for most spatial quantities we cannot represent them exactly; this can cause many spatial algorithms to be ill-conditioned.
- *Shapes are Never Exact.* All of the common methods of shape representation denote idealised shapes; any real object has a shape which is never exactly known, but instead has been manufactured or measured to some tolerance (or combination of tolerances).

[1]The classic contenders for primary representation in the solid modelling community are Constructive Solid Geometry and the Boundary Representation, but others are known and are useful.

- *Objects never have random shapes.* Many implementations of low-level geometric algorithms work well with "random" shapes, but can have problems with shapes that show a pattern. Unfortunately many objects show such patterns[2].

These problems might at first sight seem terminal, but we believe that they might be overcome by suitable *modularisation* of spatial subsystems. By using meta-level reasoning we might identify which particular spatial algorithm will perform a given task in the best manner. By hiding the representation details within a module we can write application programs that need not care which particular representation is used. By only demanding conservative answers to queries, and never "exact" answers, we can leave the lowest level modules the problem of deciding what sort of arithmetic is required, and what sorts of tolerances.

At present we have little hard evidence for the usefulness of this view. However we are currently implementing a system designed with this modular philosophy in mind, namely the spatial reasoning components for the Oxford Autonomous Guided Vehicle research.

3 The Oxford Autonomous Guided Vehicle Project

The Oxford Autonomous Guided Vehicle Project is a serious attempt to integrate many recent advances in sensing and spa' 'al planning to provide a reliable system that can operate in a semi-structured (factory) environment. The is using a research version of the a vehicle developed and manufactured jointly by GEC plc (in the UK) and the Caterpillar Corporation (in the USA). This version has the same guidance, control and sensing capabilities of the standard vehicle, but it is smaller (more suitable for our laboratory) and is built to allow the fitting of extra equipment. To deal with the uncertainties in the world we must detect them, and so a number of different sensor systems are being attached to the vehicle, including vision cameras, a sonar array, a depth sensor, and an infra-red sensor. These sensor systems are major research projects in their own right (see [5] for an overview). The reason for having a number of different sensors is to be able to combine their output, with the noisy or poor data from one sensor being made up by better information from others. This is the domain of *sensor data fusion*, which is another major research topic. From the point of view of this paper the sensor data fusion system forms a convenient bridge between the sensor systems themselves, and the geometric models which define the *planning* systems' model of the world. This role is highlighted by the overall system architecture of the project (figure 1).

Effectively information flows *from* the sensors to the sensor data fusion stage, and the sensor data fusion stage updates the world model with information it regards as pertinent and reliable. This architecture effectively allows the spatial planning systems to operate on the assumption that the information from the sensing systems is perfect; we are ignoring any tolerances in the data. (This is excusable in the application domain, but might not be, of course, in other domains.)

Two planning modules are shown connected to the world model. These correspond to two basic modes of the vehicles operation, namely motion between start and goal points whilst avoiding obstacles (as detected by the sensing systems), and acquisition of a pallet from a pile of mixed pallets and boxes. These operations form an important subset of the operations performed by typical factory vehicles. These modes were chosen for a particular reason: the first mode involves planning a path that avoids objects, whereas the second mode involves planning a path

[2]An example which we know well is the computation of configuration space obstacles by the vertex-set difference method [19]; this produces a regular set of points which broke most of the convex hull algorithms that we fed it to.

that contacts objects (*viz.,* picking up pallets using a fork-lift attachment). The support of these two modes within the *same* system is a major challenge for our modular approach. The final module—the overall planner—is a relatively simple task planner that selects a subtask using information supplied by a central factory computer. (Figure 2 shows a model of our vehicle approaching a pallet in our laboratory.)

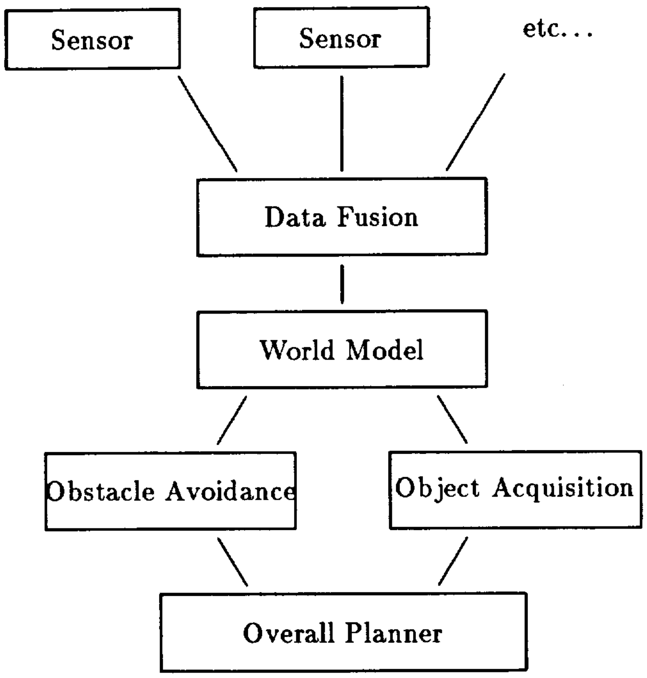

Figure 1: Overall System Architecture

In fact each of the three spatial reasoning modules (the world model, the path planner, and the acquisition module) are themselves modular, as discussed below.

3.1 The World Model

The world model accepts requests for information from the two spatial planning modules, and uses a combination of four internal models to answer the requests. This view of the world model is sketched in figure 3.

The four components of the world model can be divided into two pairs, consisting of static and dynamic information. The factory layout model "looks" like a two-dimensional plan of the factory, on which are marked static items (e.g., machining centres, pillars, doorways), quasi-static items (e.g., waste bins, doors), and nominal roadways. The 3D models are three-dimensional representations of objects that the vehicle senses or (literally) comes into contact with, for which a simplified two-dimensional projection will not suffice. (If there are many instances in the factory of, say, parts bins, only a single instance is stored in this component.) This split between two-dimensional and three-dimensional information is there partly because lends itself to the makings of an efficient system, but mainly because it is natural: factory layouts are a common representation (and are good for planning routes), and three-dimensional databases are generally used for holding *instances* of objects. When the sensing systems require information

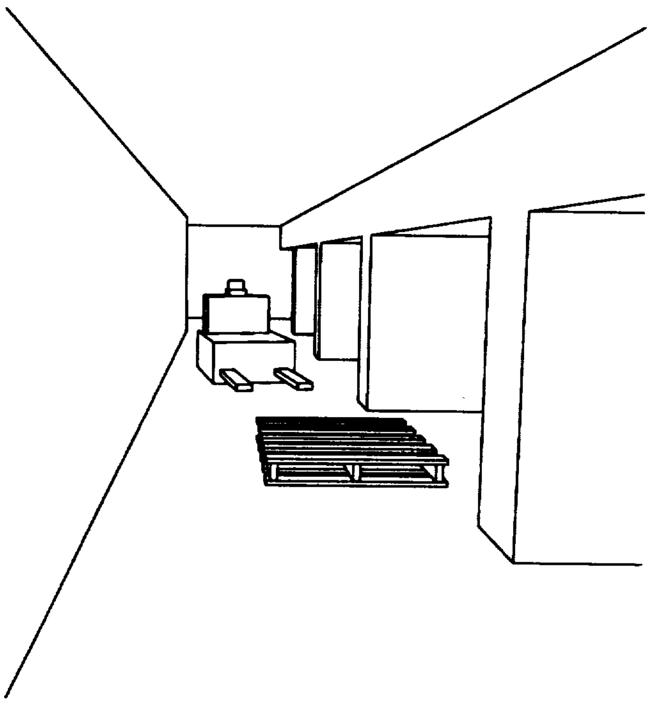

Figure 2: A ROBMOD Model of the AGV Laboratory

about what is visible, the kernel should refer to the layout model to discover which objects are (potentially) within the view, make up a three-dimensional model using instances from the solid modeller, and extract the visibility information from that.

The other components of the world model will change, both due to the discovery of unexpected objects and due to the movement of the vehicle itself.

3.2 Obstacle Avoidance

The fact that the environment of the AGV is reasonably well-structured means that we can take advantage of very simple path planning algorithms; in particular, much of the time the AGV can use generate-and-test, whereby a path is proposed and then checked for validity. In turn, proposing paths for the AGV is normally quite simple, as unless there are reasons to do otherwise the vehicle can just uses the factory roadways. The only real problem occurs when an unexpected obstacle is encountered, when we expect one of three strategies to be used:

- If the obstacle is small we will use a potential-field approach to attempt to define a detour motion around it [16]; this motion is verified by the path-checker before being accepted.
- If the obstacle is larger the system will use a C-space approach, using a number of two-dimensional C-space maps covering a small number of vehicle orientations [19,18].
- If the route is blocked the vehicle will try to backtrack to find another route.

To perform collision detection we will use the routines already built into the ROBMOD system [10,9]. These routines have been optimised to perform intersection tests using S-bounds, which is a simple method for reasoning about the bounding volumes [8].

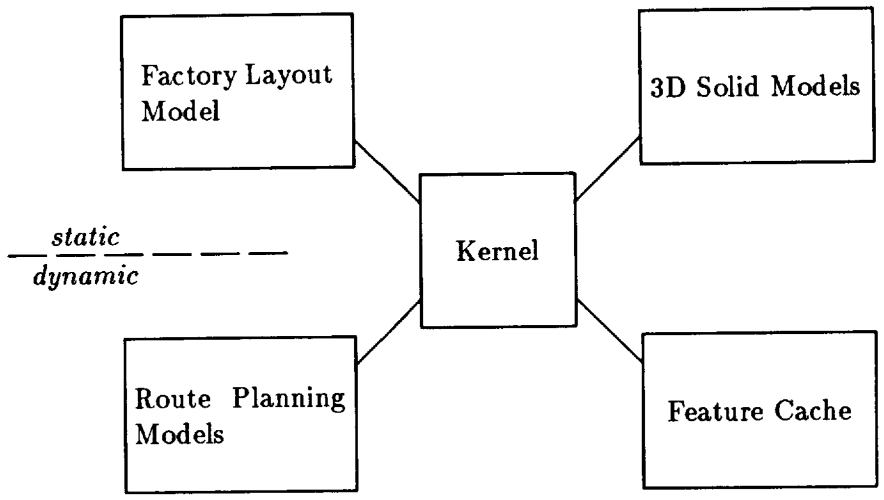

Figure 3: World Model Components

3.3 Object Acquisition

The purpose of the object acquisition experiment is to introduce the AGV into a space into which a number of loaded pallets have been positioned in an irregular manner. The AGV will have a fork-lift attachment, and has to identify the pallets, compute their orientations, and plan how the acquire the pallets using the fork-lift. In doing so it must take into account the positions of other objects and pallets in the area in order to avoid collisions. The path planning required in this case is thus of a different calibre from that required for obstacle avoidance, as it is necessary for the forks of the vehicle to come into close proximity with other objects. However, the class of objects that has to be tackled is restricted—namely, in the first instance, to pallets. Thus our approach is to use simple skeletonised plans to propose paths for the vehicle, which are then tested for validity. This will clearly work in simple cases; the challenge will come in getting the system to work well in relatively cluttered cases.

4 Related Work

Most of the push for what we now call geometric or solid modelling came from the engineering community [6,28,29]; [2] is an exception it that it was originally intended for vision research, although it was never really used for that purpose.

Collision avoidance has long been of interest in the robotics community [25,30]; more recently the configuration space approaches have been popular [21,19,33], although other methods are known [24,16]. No general methods for grasp planning have yet been developed [32,23]. Interference detection has been well-studied [3] and many methods for performing collision detection are known [10,12]. Mechanical assembly design is the study of the Design to Product project [14,27].

$$\begin{tikzcd}
\textbf{Artificial Geometry} \arrow[d] \arrow[dd, bend left] \\
\textbf{Spatial Reasoning} \arrow[d] \\
\textbf{Computational Geometry} \arrow[d] \\
\textbf{Geometric Programming}
\end{tikzcd}$$

Figure 4: The World of Spatial Reasoning.

5 Looking Ahead

Although the implementation work for the AGV project is still in its infancy, we can see a potential problem ahead if we were to try to extend our strict, hierarchal model of spatial reasoning to more complex problems. Effectively, our current model has three levels. At the top level there is the spatial reasoning 'module', that given a well-posed problem selects a suitable algorithm to solve it. The next level down is the realm of computational geometry; the algorithms themselves. These algorithms have to be implemented on real computers, and so there is a final level where these algorithms are converted into suitable code—I call this the geometric programming level, and it is currently handled by humans. The potential problem is that exemplified by the fact that many computational geometry algorithms will break on certain inputs. If this happens with our strict hierarchy there is no mechanism to report useful information back. The eventual solution to this problem is unclear; I postulate an overseer level that sits on top of the existing hierarchy and can sense when things are going badly. With tongue in cheek I have dubbed this extra level the "Artificial Geometry" level (figure 4).

Taking the idea of an artificial geometry expert one stage further, we could envisage such an overseer that could write the computational algorithms for itself; at present only humans can do this task, which is more of an art than a science. Of course, we have yet very little idea as to how we could construct such an "expert"; however, we are keeping its future existence in mind as we tackle some of the very difficult problems on the lower levels of the spatial reasoning hierarchy.

Acknowledgements

Many of the ideas in this paper took form whilst the author was employed under the McDonnell Douglas Independent Research and Development Programme. Karl Kempf and Bob Culley were responsible for encouraging my interest in spatial reasoning, and provided much of the inspiration (and much of the legwork) for this work. Also thanks to Mike Brady, and others in the Oxford Robotics Group, for feedback on these ideas. The AGV project is supported by the ACME Directorate of the SERC, and the author was supported by an SERC Atlas Research

Fellowship during the early stages of that programme. We owe thanks to our supporters on the AGV project, both from academia and elsewhere, and especially to GEC-FAST for providing us with the test vehicle.

References

[1] A. P. Ambler, S. A. Cameron, and D. F. Corner. Augmenting the RAPT robot language. In U. Rembold and K. Hormann, editors, *Languages for Sensor-Based Control in Robotics*, pages 305–316, Springer-Verlag, ref. F29 (1987), Castelvecchio Pascoli, September 1986. Also as University of Edinburgh DAI Research Paper 330.

[2] B. G. Baumgart. *Geometric Modelling for Computer Vision*. PhD thesis, Stanford, October 1974. Reprinted as Stanford AIM-249.

[3] J. W. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1), 1979.

[4] J. M. Brady. Artificial intelligence and robotics. *Artificial Intelligence*, 26(1):79–121, April 1985.

[5] Michael Brady, Stephen Cameron, Hugh Durrant-Whyte, Margaret Fleck, David Forsyth, Alison Noble, and Ian Page. Progress towards a system that can acquire pallets and clean warehouses. In *Fourth Int. Symp. Robotics Research*, pages 359–374, Santa Cruz, August 1987.

[6] I. C. Braid. *Designing with Volumes*. PhD thesis, University of Cambridge (U.K.), 1973.

[7] R. A. Brooks. Symbolic error analysis and robot planning. *Int. J. Robotics Research*, 1(4):29–68, Winter 1982.

[8] S. A. Cameron. Efficient intersection tests for objects defined constructively. *Int. J. Robotics Research*, 8(1), February 1989. Similar to Oxford Robotics Group OU-RRG-87-1.

[9] S. A. Cameron. *Modelling Solids in Motion*. PhD thesis, University of Edinburgh, 1984. Available from the Department of Artificial Intelligence.

[10] S. A. Cameron. A study of the clash detection problem in robotics. In *IEEE Int. Conf. Robotics and Automation*, pages 488–493, St. Louis, March 1985.

[11] Stephen Cameron and Jon Aylett. ROBMOD: a geometry engine for robotics. In *IEEE Int. Conf. Robotics and Automation*, pages 880–885, Philadelphia, April 1988.

[12] John Canny. On detecting collisions between moving polyhedra. *IEEE Pattern Analysis and Machine Intelligence*, 8(2):200–209, March 1986.

[13] R. Chatila. Mobile robot navigation: space modelling and decisional processes. In *3rd Int, Sym. Rob. Res.*, Gouvieux, 1985.

[14] Paul Fehrenbach and Tim Smithers. Design and sensor-based robotic assembly in the "design to product" project. In *Fourth Int. Symp. Robotics Research*, Santa Cruz, August 1987.

[15] E. Kant. Understanding and automating algorithm design. In *9th IJCAI*, Los Angeles, 1985.

[16] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Int. Conf. Robotics and Automation*, pages 500–505, St. Louis, March 1985.

[17] L. I. Liebermann and M. A. Wesley. AUTOPASS: an automatic programming system for computer-controlled mechanical assembly. *IBM Journal of Research and Development*, 21:321–333, July 1977.

[18] T. Lozano-Pérez. Motion planning for simple robot manipulators. In *3rd Int. Sym. Rob. Res.*, Gouvieux, 1985.

[19] T. Lozano-Pérez. Spatial planning—a configuration space approach. *IEEE Transactions on Computers*, 108–120, February 1983.

[20] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. In *First Int. Sym. Robotics Research*, MIT Press, 1984.

[21] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[22] J. Malik and T. O. Binford. Reasoning in time and space. In *8th IJCAI*, 1983.

[23] M. T. Mason and R. C. Brost. Automatic grasp planning: an operational space approach. In *Oxford Workshop Geometric Reasoning*, 1986.

[24] J. K. Myers. *A Supervisory Collision-Avoidance System for Robot Controllers.* Master's thesis, Carnegie-Mellon University, 1981.

[25] D. L. Pieper. *The Kinematics of Manipulators under Computer Control.* PhD thesis, Stanford, 1968.

[26] R. J. Popplestone, A. P. Ambler, and I. M. Bellos. RAPT: a language for describing assemblies. *Industrial Robot*, 5(3):131–137, 1978.

[27] R. J. Popplestone, T. Smithers, J. Corney, A. Koutsou, K. Millington, and G. Sahar. Engineering design support systems. In *1st Int. Conf. Appl. AI Eng. Prob.*, Southampton, 1986.

[28] A. A. G. Requicha. Representations for rigid solids: theory, methods and systems. *Computing Surveys*, 12(4), December 1980.

[29] A. A. G. Requicha and H. B. Voelcker. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2), March 1982.

[30] S. Udupa. Collision detection and avoidance in computer controlled manipulators. In *Int. Joint Conf. Art. Int.*, Boston, 1977.

[31] Wu Wen-tsun. Basic principles of mechanical theorem proving in geometries. *J. Sys. Sci. Math. Sci.*, 4(3), 1984.

[32] M. P. Wingham. *Planning How to Grasp Objects in Cluttered Environments.* Master's thesis, Department of Artificial Intelligence, University of Edinburgh U.K., 1977.

[33] C. K. Yap. Algorithmic motion planning. In J. T. Schwartz and C. K. Yap, editors, *Advances in Robotics 1*, Erlbaum, Hillsdale NJ, 1986.