

Geometric Reasoning for Computer Aided Design

R. R. Martin

Department of Computing Mathematics,
University of Wales College of Cardiff

1 Introduction

Geometric Computation has been widely used for over twenty years in Computer Aided Design, but until fairly recently, the emphasis has been on the end user deciding what geometric constructions to make. The main mode of use has been to treat a CAD system as being to geometry as a calculator is to arithmetic - the CAD system can perform various geometric manipulations and draw the results, but it has no built in knowledge of geometric theorems and concepts, just as a calculator has no knowledge of number theory.

For intelligent design, the next logical step appears to be for CAD systems to be able to perform *Geometric Reasoning*. What exactly geometric reasoning means, and what it is intended to do varies very much depending on who one talks to, as might be expected in a relatively new area of research. Nevertheless, there seem to be two main facets to geometric reasoning. The first of these is the ability of software to define geometric information and deduce properties of geometric objects at a high level rather than directly in terms of the details of specific geometric elements. One example of the way in which information can be processed at higher levels is feature recognition in a solid modeller, where a boss may be defined as a protrusion from the main body of a solid, irrespective of its particular geometric shape, as shown in Figure 1. Various intermediate level properties of the object such as convexity, loops of connected edges and faces, and relative sizes of faces have to be deduced from the low level geometric information, which are then combined to decide whether various highest level features are present. This must all be done in a way which is independent of the particular geometry which exists at the lowest level. A rather different example of manipulation of geometric information at a higher level is to represent all curves and surfaces as algebraic polynomial expressions. As will be explained later, this enables generic solutions to geometric problems such as finding the intersection of any two algebraic curves, rather than specific algorithms for the intersection of two straight lines, a straight line and a circle, two circles and so on.

The second facet of geometric reasoning is the ability of software to convert

one representation of a geometric object to another representation more suited to the particular task in hand. This process may involve selection and rejection of the original information, as well as deriving new information from it. For example, consider the object shown in Figure 2. When trying to deduce what weight it will support, it is natural to think of the object as being composed of four rectangular blocks supporting each other. However, when trying to decide how to manufacture the object from a solid piece of metal, it is much more useful to think of it as block minus a hole. Producing a description of the latter type is exactly what feature recognition described in the previous paragraph is attempting to do.

Given these ideas, this paper attempts to review various aspects of geometric reasoning, and to show how they might be useful for intelligent design. However, it does not try to be comprehensive in its coverage, but instead concentrates on various areas the author feels to be of major interest. One extensive area of geometric reasoning which will not be detailed in this survey is its use for image understanding. Although this is obviously of great importance for many tasks, its main uses in manufacturing are at the machining, assembly and inspection stages rather than the design stage.

Two particular publications which include papers on a wide variety of topics in geometric reasoning are [1, 2].

2 Geometry and Computer Algebra

In the introduction, computer systems were mentioned for performing arithmetical and geometrical calculations. Not surprisingly, systems have also been created and are commercially available for performing algebraic calculations. Such systems are called Computer Algebra systems. Although they originally used various heuristic methods for such tasks as evaluating indefinite integrals, the more recent trend is towards more deterministic methods often based on deep mathematical results. For a useful introduction both to the use of such systems, and the theory they are based on, see Davenport *et al.* [3].

The relevance of computer algebra methods to geometric reasoning is twofold. Firstly, restatement of a geometric problem in algebraic terms (usually polynomial equations) may mean that it can readily be solved by use of standard algebraic methods. This is a specialised case of the more general paradigm of transforming knowledge and problem statements from one representation to another which may be easier to work with than the original representation. Note that although algebraic methods are used for these computations, the results themselves can readily be converted back into geometric terms when necessary.

Secondly, and even more important, is the fact that using algebraic methods can be rather more general than solving individual geometric problems, as already mentioned in the Introduction. The example given there will be explained in further detail. In conventional CAD systems, problems like finding

the intersection of a straight line and a circle are typically solved by the software developer solving the problem geometrically on paper, deciding that it requires the solution of a particular quadratic equation, and then hard coding that solution. When using algebraic methods, the computer is now able to deal with higher level concepts such as algebraic variables and polynomials. Finding the points of intersection of *any* two curves which can be expressed in the form $f(x, y) = 0$ can be performed by the algebraic computation of the resultant of two polynomials in x and y , and using Sturm Sequences to find out how many solutions exist. The particularly useful point about doing this is that should it be desired to add ellipses to the CAD system, in the former case, special code would need to be developed to find intersections of ellipses and straight lines, and ellipses and circles, whereas if algebraic methods are used, the solution to a whole class of such problems is available.

A further, but different example of the generic nature of solutions provided by algebra systems comes when drawing a solid model using ray-tracing. In a conventional system, many individual rays must be tested for intersection with the object. Using computer algebra, it is possible to solve once where a generic ray with variable parameters intersects the model, and then substitute for the actual rays required in this general solution.

Two algebraic methods which are of particular relevance to geometric reasoning are Cylindrical Algebraic Decomposition [4, 5, 6], which allows the decomposition of an arbitrary part of n -dimensional space defined by a set of polynomial equations and inequalities into connected regions, and the construction of the adjacency relations between these regions [7]. These relations form a graph representing, for example how space is occupied by a moving mechanism, or whether various objects will fit inside another object.

The second technique is Buchberger's Algorithm for constructing Gröbner Bases [6], which are useful for attacking various problems involving systems of algebraic polynomial equations, such as deciding whether the equations are independent, finding common roots of the equations, and eliminating variables between the equations. The last computation repeatedly occurs when solving geometric problems by algebraic means, for example in turning parametric formulations of curves and surfaces into implicit ones, in solving various intersection problems and in detecting singularities.

On the practical side, Bowyer *et al.* [8] are in the progress of developing a subroutine library called GAS (Geometric Algebra System) for performing various algebraic tasks specifically required by geometry manipulating programs such as solid modellers. Their ultimate aim is to create a solid modeller which performs its computations using algebraic methods wherever possible in place of floating-point numerical methods.

A more particular use of algebraic methods has been described by Martin *et al.* [9], for finding the swept volumes of moving solid objects. Algebraic techniques are used to find what envelope surfaces are generated by each of the surfaces in the original model as it moves, while a projection based algorithm

then decides which part or parts of each of these surfaces contributes to the final swept volume. An example of the use of this computation arises in designing suitable housings for mechanisms such as a moving radar dish. A second, and perhaps rather more widespread use is in determining the volume of metal removed by a cutter as it sweeps out a path during machining in an integrated CAD/CAM system.

Todd [10] describes how algebraic methods can be used for designing mechanisms. Here, high level constraints (such as distances between points) are converted into algebraic form, and then compiled (or solved) by a computer algebra system into a conventional program. This allows various design parameters to be changed interactively while still satisfying the constraints, and the results to be displayed graphically.

A rather different use of algebraic methods is in the automatic creation of blending surfaces. Here, the sharp edge of intersection of two existing main surfaces is replaced by a small piece of blending surface joining both of the original surfaces with tangential continuity. This allows the designer to concentrate on the main features of the design, and then to automatically smooth (fillet or chamfer) the edges of these features as necessary. Various slightly different formulations of this method exist [11, 12]; the basic idea is that if A and B are the algebraic equations of two surfaces, then $(1 - u)AB + uP^2$ is a new surface which is tangent to A and B and touches them along the curve where P , a third, control surface, intersects them. The parameter u can be varied to control the range or fullness of the blend.

Finally, it should be noted that other advantages are also to be had in using methods from computer algebra systems. For example, by using rational arithmetic it is possible to avoid some of the numerical instabilities which arise when performing real arithmetic with computer limited precision. Nevertheless, as is often the case, there is a tradeoff. The gain in precision is obtained at the expense of the size of the rational coefficients, which must be represented internally using multiple precision arithmetic, with many tens if not hundreds of decimal digits. Indeed, a well-known general problem with computer algebra is *intermediate expression swell*, where although the input expressions and the output expression may be just a few lines long, intermediate expressions can grow exponentially in length before collapsing back to the size of the final result. Current algebra systems are for this reason heavy users of memory, and computer time.

3 Automated Theorem Proving

When reasoning about geometry, the high level concepts under consideration eventually need to be broken down into lower level queries to solve problems. Sometimes this can be done directly. For example, telling whether one convex polyhedral object is inside or above another can be done by comparing their

vertices. Again, however, particular code must be written to solve each such specific problem. In the spirit of Geometric Reasoning working at a higher level, there exist techniques of *Automatic Theorem Proving* where a query about an object is expressed as a theorem, and code which can check *any* such geometric theorem is used to answer the query. It is not too difficult to see the parallel with Expert Systems - in that case a general purpose inference engine works from a set of stored facts and input data describing a particular problem towards a goal, while in this case a general purpose theorem prover works from a particular set of facts conjectures towards a proof.

A powerful and general algebraic method for proving geometric theorems has been given by Wu [13, 14], and has been implemented with considerable success by various people, including Wu himself and Chou [15]. The method basically works by writing the given facts and conjectures as polynomials, where the variables in these polynomials are coordinates of points. The goal is then to find whether the conjecture follows from the given facts. This is so if the common zeros of the given statements are also zeros of the conjectures. Various methods may be used to find these common zeros, Wu's original idea being based upon pseudo-division and polynomial factorisation, while another possibility relies on the Gröbner Basis method described above [16]. Wu's method may also be extended to describe results which involve inequalities, as well as equations, by introducing extra variables.

One advantage of Wu's technique is that it solves problems generically, in other words, any theorems proved will still be true in degenerate cases such as ones where distances between points becomes zero, lines are accidentally parallel, circles reduce to a single point, and so on. This is an important advantage, as enumerating and handling special cases with special code is a tedious and error prone concern for most current CAD systems. Nevertheless, if it is wished to rule out specific degenerate cases from the proof (X is true except when Y), this can be done by adding extra equations describing the exclusions.

It should also be noted that other theorem proving methods have been suggested, such as the method due to Kapur [17], which uses similar ideas to Wu's, but instead uses proof by contradiction, and tries to prove that the negation of the conjecture is inconsistent with the given facts. An interesting extension which Kapur's method can handle is as follows: given a set of initial facts which do not lead to the desired conclusion, then it can find additional facts, if they exist, which are consistent with the original facts, and which when added to the original set of facts, does cause the set of facts to imply the desired conclusion. These additional facts often turn out to be extra constraints, *e.g.* that certain points must be distinct.

It is not difficult to see that theorem proving has many potential applications for design, for example when checking that a set of geometric constraints on a design lead to other desired geometric properties. However, the complexity of the algebraic problem to be solved would seem to grow rapidly with the geometric complexity of a design, and whether Wu's method can be practically

useful still has yet to be determined. Although simple theorems with a small number of variables can be solved in a few seconds, the rather more complex problems posed in design may well take up to several tens of hours, or more.

A second problem with theorem proving systems is that they work in the field of complex numbers rather than the reals. Although complex numbers are more general than the reals, loosely speaking, and so any theorem which is true for complex numbers must include the same theorem for real numbers as a special case, the reverse of this is unfortunately not true. Thus, a theorem may be true for the reals, but the theorem prover may not be able to prove it, because it is not true for complex numbers. Taking a concrete example, for real numbers the given fact $x^2 + y^2 = 0$ implies that the theorem $x = 0$ is also true, but this is not the case if x and y are allowed to be complex, as shown by the pair $x = i, y = 1$. The implications of this observation for solving practical problems of real geometry are not yet well understood.

4 Reasoning about User Input

Although many current CAD systems allow geometric input, they only do so in a straightforward way, where geometric shapes drawn represent themselves. Geisow [18] contends that diagrams can be used for much wider purposes, as often they can be more readily understood than textual descriptions. While this observation has been widely used for output, its uses for input have been quite limited up to the present. Here, geometric reasoning will be an essential tool for capturing designer's intentions presented in schematic diagrams. Geisow proposes a twofold approach which first builds up a structured diagram from the primitive graphical elements (a syntactical step), followed by mapping of the structured diagram to the particular application domain (a semantic step). As he points out, this will allow modular software construction from reusable components. Related work in this area has also been carried out by Pereira [19] and Arya [20].

Another area which is attracting current interest is sketch input to CAD systems. Here, artists' sketches are used as a starting point for automatic derivation of a CAD model, as opposed to the more usual methods of precise user input of desired geometric elements. Such sketches are by their nature incomplete, and unreliable in terms of both linear and angular dimensions. Here, geometric reasoning is needed to infer the artists' intentions from the sketch. Suffell *et al.* [21] note that at different phases of the design, different amounts of detail are required, allowing further information to be added to that provided by sketches at a later stage. They advocate the use of computational stereo techniques for matching information provided in multiple sketches, but note that allowing for differences in sketches due to "artistic licence" may be problematical.

Fisher *et al.* [22] note that geometric consistency and merging operations need to be performed when information is obtained from several different fea-

tures in the two views. They describe an extension of *ACRONYM* [23] in which such information is expressed as constraints, which can be manipulated algebraically. These constraints are associated with object hypotheses to form networks which must be forced into consistency. Fisher *et al.* expect fast convergence of such methods, especially when executed in parallel.

Indeed, it is not difficult to see that many of the various techniques proposed for geometric reasoning in the field of computer vision are of potential use in interpreting sketched input, or even for using photographs as a basis for design.

Another use of geometric reasoning in input lies in enabling the user to identify higher levels of a solid object than just its faces, edges and vertices. He may wish, for example, to move the position of a hole or pocket, a composite item made of many simpler geometric elements, within the object. For such reasons, and also for uses such as automatic generation of process plans [24] and part codes [25], there has been much interest in the use of geometric reasoning to identify geometric features in solid objects.

In another approach to the problem of input, Martin [26] points out that most current CSG solid modelling systems use relational algebra to define shapes in terms of set operators which explicitly combine primitive shapes. An alternative is to use relational calculus instead to define a shape in terms of the points it must contain. The former is prescriptive (do this, do that), and is well suited to traditional languages like *PASCAL*, whereas the latter is descriptive (the result required has these properties), and is much more suited to declarative languages like *PROLOG*. Although it may be more natural for users to use the former for input, the latter is probably more suitable as an internal representation for geometric reasoning modules. Importantly, relational algebra and relational calculus are equivalent, and can readily be converted from one form into the other [27].

The problem of feature recognition may be solved in part by suitable choice of relational algebra operators, especially if they correspond to particular manufacturing operations, as some features can then be captured at the input stage. However, as Jared [28] points out, different geometrical aspects of a solid object may well be regarded as features at different stages of design and production. Designers tend to think in terms of functional features while process planners require manufacturing features (Husbands *et al.* [29]). A further observation made by these authors is that interrelationships between the features are often more important than the isolated features themselves. As they point out, feature input by the designer is a time consuming process which urgently needs to be replaced by automatic feature recognition.

For an excellent discussion of the whole area of automating feature recognition, see the speculative paper by Woodwark [30]. In particular he notes that most approaches, perhaps apart from Woo's [31], are based on boundary models, but this can lead to both explicitly storing information which is not required in such detail, and to expensive global computations to ensure that other regions of the component do not invalidate a feature (a bolt hole whose opening

is blocked by another part of the object is of no use as a bolt hole). He instead proposes a method of feature recognition based on set theoretic models, despite many problems of using this approach.

5 Expert Systems

Because design is often based on heuristics as well as scientific reasoning, it is not surprising that the use of expert systems has often been proposed as a suitable method for storing these heuristic rules. However, although expert systems are well suited to *logical* reasoning, and possibly *statistical* reasoning when fuzzy logic and similar methods are used, in general it would seem that present expert systems probably do not have the necessary mathematical tools (such as algebraic manipulation capabilities) for the more complex *geometric* reasoning tasks. Nevertheless, several attempts have been made to perform geometric reasoning tasks using expert systems.

For example, as Murray *et al.* point out [32], various relationships exist between the different components of an assembly. If geometric features of one part of an assembly are changed, it is often necessary to make changes to other parts of the assembly as a result. Thus, if the diameter of a shaft is increased, the diameter of any bearings supporting that shaft will also need to be increased. One way of allowing this type of change is to rely heavily on standardised parts whose geometric shape is specified by a set of parameters. Murray suggests the use of an expert system for keeping track of the geometric relationships between the parts of an assembly. Firstly, facts and rules about the design are entered, and then in a second step, parameters are assigned to create a specific instantiation of the assembly, where the facts and rules ensure that all of the design constraints are adhered to. Note that alternative designs may be created not only by changing the parametric values, but also by modifying the design rules themselves. Typical rules allow for “part of” and “type of” relationships, while the user may also define higher level rules of his own, such as what “force-fitted” means in geometric terms.

Let us go back to the earlier remarks made about the general suitability of expert systems for performing geometric reasoning. If an expert system were to be interfaced to a conventional solid modeller, such a system could offer the reasoning capabilities of the expert system, while using the geometric modeller to solve any geometric problems posed. One possible area the current author has identified where this approach may be of some use is in checking the conformance of a design with standards of a geometric nature. One example might be whether the correct shape and number of brackets have been used to support a component. It is not too difficult to further envision the use of automatic finite element mesh generation from solid models, as described by Wordenweber [33], to answer further questions asked by the expert system about strength and other properties. It is hoped that a future project will tell us

whether a system where the geometry and the reasoning are only loosely coupled can be effective for solving problems of this type, or whether an integrated system is necessary for solving geometric reasoning problems.

Another type of task where expert systems and CAD systems have already been linked is in the area of ergonomic design. Bonney *et al.* [34] describe the use of the *SAMMIE* CAD system and *ALFIE* expert system in this field. The layout of workplaces or equipment can be checked for a variety of requirements, from simple ones of whether everything will fit, to the more complex criteria involved in deciding whether an operator can see and easily operate all of the controls. A particular requirement and feature of their system is the ability to make geometric models of human beings.

6 Other Possibilities

Although the main part of the paper has concentrated on various major aspects of Geometric Reasoning which the author feels are particularly relevant to design, many other possibilities are also being actively pursued. In this final section, a brief summary of some of the more interesting ones will be given.

The use of networks of constraints has already been mentioned with respect to input above. Another area where very similar ideas are of use is in the automatic analysis of geometric tolerances, and Fleming [35] suggests a method of doing this based upon representing tolerances by a network of datum values and tolerance zones.

Other work for representing uncertain geometry has been carried out by Durrant-Whyte [36], principally for reasoning in the area of robotics, but it would also seem that his ideas might well be of use in representing tolerances in design. He particularly considers the problems of how different uncertainty measures may be combined. To do so, he represents uncertain points, lines and surfaces as stochastic point processes described as a probability distribution on the parameter space describing the underlying object. The manipulation of uncertainty measures can then be performed by transforming and combining these probability measures.

An interesting idea due to Canny [37] links together algebraic ideas with solid modelling ideas. In his parlance, a semi-algebraic set is what CAD practitioners would call a CSG (Computational Solid Geometry) solid model. He gives an algebraic procedure for constructing a “Roadmap” of a semi-algebraic set, which is a one dimensional structure, such that each connected component of the original model corresponds to a single connected component of roadmap. Furthermore, it is possible given a point in the original set to rapidly find a corresponding point in the roadmap, and as a corollary, decide quickly if two given points are in the same component of the original. Although he originally proposed these ideas in the context of robot path planning (where existence of a path between two points depends on whether the starting and end point are

in the same component of a set in configuration space), it is obvious that this technique has much wider potential use in design. A reasoning module may wish to tell whether two points are in the same component of an assembly or mechanism when deciding how moving one point will affect the other. More exotic possibilities include deciding whether a geometric design satisfies certain basic requirements of vibrational or electrical insulation by testing whether certain points are in the same or different components of the model.

Although one advantage of algebraic systems is their immunity to numerical problems, this is offset by their relative inefficiency, and so alternative approaches are also being studied to the accuracy problem. One example is the work by Milenkovic [38] on how verifiably correct geometric algorithms can be constructed, which are still based on limited precision arithmetic. He proposes two methods, the first of which, data normalisation, transforms the geometric structure of the problem into a configuration where all limited precision calculations give correct answers. The second method, the hidden variable method, constructs configurations which belong to infinite precision objects, but without explicitly representing them.

Another rapidly expanding area of great significance is theoretical computational geometry. Many efficient data structures and algorithm design techniques have been devised for reasoning about geometric structures. An excellent tutorial on this subject is provided by Guibas *et al.* [39]. One current problem, highlighted by this paper, is that the resulting algorithms proposed by such research are usually quite subtle and extremely complex, and relatively few of them have been implemented in practice.

As a final statement, the need for hierarchical representations of geometric information and the combination of different reasoning methods at different levels should be noted, a point made very clearly by Barry *et al.* [40]. To a certain extent, current boundary representation modellers do this already by separating their data into the topological and geometrical components, where restructuring problems are abstracted from particular geometric computations. Barry *et al.* take these ideas further. They propose a model which stores both explicit geometric (and topological) data, and logical relations about geometric entities which are used to control numerical accuracy problems. For example, a given vertex may be computed perhaps by the intersection of any two edges meeting at it, or as the intersection of any three faces. Depending exactly on which of these possible definitions is used to find the vertex, slightly different numerical results may be obtained. The logical relations are used to merge these possibilities, by perturbing the geometric entities to ensure that any geometric computation which leads to the same result gives the same geometric answer. Nevertheless, incorrect logical decisions can be made when, for example, two points are accidentally close together, and are determined to be logically the same point. A higher reasoning level is then needed to keep track of which geometric entities should and should not be logically associated.

As has been pointed out above, algebraic and theorem proving methods

can consume large amounts of computer time, so multi-level reasoning systems which decide when and how to use these powerful tools to best effect will be necessary.

References

- [1] D. Kapur, J. L. Mundy (Editors). *Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986*. Artificial Intelligence, Special Issue, To Appear.
- [2] J. R. Woodward (Editor). *Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre)*. Oxford University Press, 1989.
- [3] J. H. Davenport, Y. Siret, E. Tournier. *Computer Algebra Systems and Algorithms for Algebraic Computation*. Academic Press, London, 1988.
- [4] D. S. Arnon, G. E. Collins, S. McCallum. *Cylindrical Algebraic Decomposition I: The Basic Algorithm*. SIAM Journal of Computation **13** 865-877, 1984.
- [5] D. S. Arnon, G. E. Collins, S. McCallum. *Cylindrical Algebraic Decomposition II: An Adjacent Algorithm for the Plane*. SIAM Journal of Computation **13** 878-889, 1984.
- [6] B. Buchberger, G. E. Collins, B. Kutzler. *Algebraic Methods for Geometric Reasoning* Ann. Rev. Comput. Sci., 3, 85-119, 1988.
- [7] J. H. Davenport. *Robot Motion Planning*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodward, Oxford University Press, 1989.
- [8] A. Bowyer, J. Davenport, P. Milne, J. Padget, A. F. Wallis. *A Geometric Algebra System*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodward, Oxford University Press, 1989.
- [9] R. R. Martin, P. C. Stephenson. *Swept Volumes in Solid Modellers*. In: Mathematics of Surfaces III, Ed. D. C. Handscomb, Oxford University Press, 1989.
- [10] S. J. P. Todd. *Programming Interactions by Constraints*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodward, Oxford University Press, 1989.
- [11] C. Hoffmann, J. Hopcroft. *Quadratic Blending Surfaces*. Computer Aided Design, **18**, 1986.

- [12] A. Middleditch, K. Sears *Blend Surfaces for Set Theoretic Modelling Systems*. ACM SIGGRAPH Computer Graphics, **19**, 161-170, 1985.
- [13] D. Kapur, J. Mundy *Wu's Method: An Informal Introduction* In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.
- [14] W. Wu *On the Decision Problem and the Mechanisation of Theorem Proving in Elementary Geometry*. Scientia Sinica, **21**, 150-172, 1978.
- [15] S.-C. Chou. *Proving Elementary Geometry Theorems Using Wu's Algorithm* In: Theorem Proving: After 25 Years, Ed. Bledsoe, Loveland, Contemporary Mathematics **29**, 1984.
- [16] B. Kutzler, S. Sifter. *Automated Geometry Theorem Proving Using Buchberger's Algorithm*. 1986 Symposium on Symbolic and Algebraic Computation (SYMSAC 86), Waterloo, Canada, 1986.
- [17] D. Kapur, J. Mundy *A Refutational Approach to Geometry Theorem Proving*. In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.
- [18] A. Geisow. Recognition and Generation of Symbolic Diagrams. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [19] F. C. N. Pereira. *Can Drawing be Liberated from the Von Neumann Style?* In: Logic Programming and its Applications, Ed. M. van Caneghem and D. H. D. Warren, Ablex, 1986.
- [20] K. Arya. *A Functional Approach to Picture Manipulation*. Computer Graphics Forum, **3**, 35-46, 1984.
- [21] C. Suffell, G. N. Blount. *Sketch Form Data Input for Engineering Component Definition*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [22] R. B. Fisher, M. J. L. Orr. Geometric Constraints from $2\frac{1}{2}$ D Sketch Data and Object Models. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [23] R. A. Brooks. *Symbolic Reasoning Among 3D Models and 2D Images*. Artificial Intelligence, **17**, 285-348, 1981.

- [24] G. E. M. Jared. *Shape Features in Geometric Modelling*. In: Solid Modeling by Computers, Ed. M. S. Pickett, J. W. Boyse, Plenum, 1984.
- [25] L. K. Kyprianou. *Shape Classification in Computer Aided Design*. Ph.D. Thesis, University of Cambridge, 1980.
- [26] R. R. Martin, D. I. Howells *Relational Algebra, Relational Calculus and Computational Solid Geometry*. In: Theoretical Foundations of Computer Graphics and CAD, Ed. R. A. Earnshaw, Springer, 1988.
- [27] P. M. D. Gray. *Logic, Algebra and Databases*. Ellis Horwood, 1984.
- [28] G. E. M. Jared. *Recognising and Using Geometric Features*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [29] P. Husbands, F. Mill, S. Warrington. *Part Representation in Process Planning for Complex Components*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [30] J. R. Woodwark. *Some Speculations on Feature Recognition* In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.
- [31] T. C.-H. Woo. *Computer Understanding of Design*. Ph.D. Thesis, University of Illinois, 1975.
- [32] J. L. Murray, M. H. Miller *Knowledge-Based Systems in Process Planning and Assembly Design* In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [33] B. Wordenweber *Automatic Mesh Generation of Two and Three Dimensional Curvilinear Manifolds*. Ph.D. Thesis, University of Cambridge, 1981.
- [34] M. Bonney, N. Taylor, K. Case *Using CAD and Expert Systems for Human Workplace Design* In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.
- [35] A. D. Fleming. *A Representation of Geometrically Toleranced Parts*. In: Geometric Reasoning (Proceedings of a Conference held at the IBM UK Scientific Centre), Ed. J. Woodwark, Oxford University Press, 1989.

- [36] H. F. Durrant-Whyte *Concerning Uncertain Geometry in Robotics* In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.
- [37] J. Canny *Constructing Roadmaps of Semi-algebraic Sets, Part I: Completeness* In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.
- [38] V. J. Milenkovic *Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic* In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.
- [39] L. J. Guibas, J. Stolfi. *Ruler, Compass, and Computer: The Design and Analysis of Geometric Algorithms*. In: Theoretical Foundations of Computer Graphics and CAD, Ed. R. A. Earnshaw, Springer, 1988.
- [40] M. Barry, D. Cyrluk, D. Kapur, J. Mundy *A Multi-Level Geometric Reasoning System for Vision* In: Geometric Reasoning - Contents Selected From a Workshop on Geometric Reasoning at Keble College Oxford, July 1986, Ed. D. Kapur, J. Mundy, Artificial Intelligence, Special Issue, To Appear.