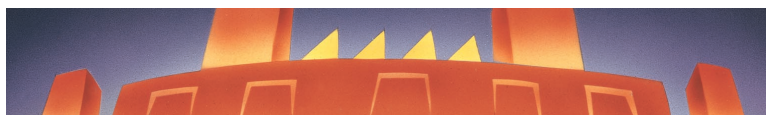


Aristides A. G. Requicha



Geometric Reasoning for Intelligent Manufacturing

A blend of artificial intelligence techniques and geometric computation can tackle the difficulties in automated manufacturing of reasoning about the geometry of products.

Manufacturing automation is at least as old as the industrial revolution, which brought about mass production and dedicated, fixed-task machinery. Flexible automation and the associated variable-task, programmable tools date from the 1950s, with the introduction of numerically controlled (NC) machining and computer-aided design (CAD).

A modern, high-level view of the automation field can be summarized as follows: At the center are representations, or computer models, of products, such as an assembly of electromechanical components; processes, such as a sequence of machining operations; and resources, such as machine tools or robots. Around this center are the activities that create and use the representations.

Creating product representations is usually called design, or CAD when emphasizing its computational aspects. In industrial practice, a design for a mechanical part or assembly is specified by its geometry plus a few nongeometric properties, such as material or hardness. Geometry is traditionally conveyed through engineering drawings. CAD helped replace such drawings, first with wireframe models, or unorganized sets of object edges, and later with solid models. Only solid models are unambiguous, completely defining the shape of objects; other models are ambiguous. Therefore, solid models are suitable sources of data for programs that automatically

answer such queries as: "What is the volume of this object?" and "Do these two objects collide?" The evolution of solid modeling has been documented in the literature [7, 8, 9, 10]. However, the conversion to solid modeling is far from complete; drawings and wireframes are still widely used in industry.

Today, human users define a product's geometry directly. In the future, mechanical design is likely to follow in the steps of very large-scale integration (VLSI) design and specify products at a higher, functional level. Geometry will then be inferred through the analogs of silicon compilers. We are still far from this stage; a major barrier is the lack of a computational characterization of design intent in terms of mechanical functions, constraints, optimization criteria, and perhaps other important properties. Design is closely linked to *analysis*. Computational methods for analyzing designs have progressed nicely, and today there are tools for computing mass properties of objects, detecting interferences, simulating motion, and so on. Progress on *synthesis* of product and process designs has been much slower.

High-level process representations are created through activities usually called *process planning* and *scheduling*. Planners reason about the geometry (and other properties) of objects, attempting to match them to the capabilities of physical processes, such as drilling or milling, and thereby generating a sequence of production operations. These operations involve fabrication, such as machining or casting; assembly; inspection; and so on. Schedulers reason about processes and resources, allocate operations to appropriate machinery, and specify the asso-

ciated timing information. Low-level representations of operations are typically imperative, taking the form of instructions to machines, as in an NC language or robot-command language. Generation of low-level instruction sequences, such as cutter paths for machining, is largely automated. But the synthesis of high-level plans is much more difficult.

We argue in this article that difficulties arise from the need to reason about space, or geometry, and that this can be done through a blend of artificial intelligence (AI) techniques and computational geometry algorithms. To support this claim, we discuss two systems developed recently at USC's Programmable Automation Laboratory. One recognizes machinable features in a product's model, the other generates high-level process plans for dimensional inspection. Work on spatial reasoning for assembly is covered in [6] and in other sources.

Resource representations for machines, factories, and so on are mostly created manually but may in the future be generated automatically by facility planners that must reason about plans, schedules, timing, size of orders, and various business and economic factors.

All of these design and manufacturing activities must be integrated among themselves and with management and business systems, which are also important but beyond the scope of this article. Integration implies standards and shared ontologies. According to the tenets of modern concurrent engineering, many of these activities should occur simultaneously, so that, for example, manufacturing considerations can influence product design. Simultaneous activity requires communication and negotiation—the hallmarks of cooperative work.

Underlying modern design and manufacturing systems is a computational infrastructure involving computer hardware, networks, databases, collaboration software, user interfaces, graphics, and more. These areas are not specific to automation—

they cover much of the computer science discipline—but design and manufacturing disciplines generate numerous challenging problems.

Recognition of Machinable Features

Process planning for machined parts typically requires reasoning about machining features, such as holes, slots, and pockets. Given a part to be fabricated and the stock or raw material, the set difference stock – part, usually called the *delta volume*, is the material that must be removed by machining operations. Machining features are volumes, or solids, that can be removed by individual operations. Feature recognition amounts to decomposing the delta volume into smaller volumes that can be associated with machining operations. Figure 1 shows a simple industrial part to be cut from a parallelepiped of metal that just encloses the part and the features recognized by USC's Integrated Incremental Feature Finder (IF²) system. One can think of delta-volume decomposition as a technique for computing *differences* between the goal and start states in a means-ends AI problem-solving procedure.¹ The differences may be *reduced* by applying the relevant operators, which correspond to machining operations.

Automatic feature recognition is a challenging problem; many solutions have been proposed in the literature [14]. A popular approach consists of defining features as graph-structured patterns of faces and edges, then searching for these patterns in the boundary of an object. The search involves subgraph isomorphism detection, which is NP-complete. The graph-search approach has difficulties dealing with features that interact volumetrically, because interactions may damage the patterns beyond recognition; see Figure 2 for a simple object with interacting features recognized by IF².

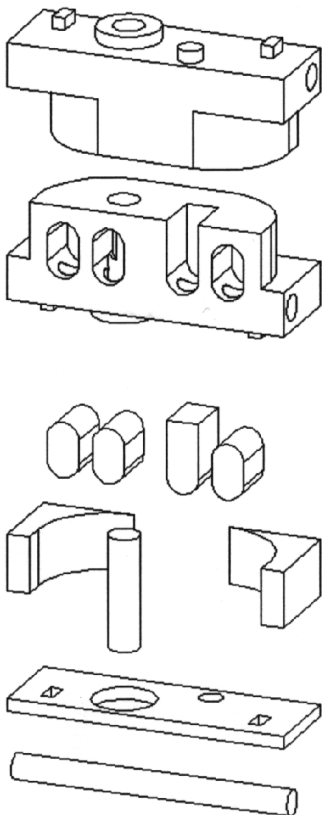
At USC, we built two recognizers: the Object Oriented Feature Finder (OOFF) [14, 15] and its successor IF² [2, 3]—both capable of recognizing interacting features while having other interesting characteristics. Here, we focus on IF², which subsumes much of OOFF and adds to it capabilities for incremental operation concurrent with CAD and for exploitation of information sources beyond the nominal geometry of parts.

IF² collects *evidences* from various sources:

- Face patterns in a part's boundary;
- Tolerances (e.g., parallelism between two faces is a clue for a slot);
- Attributes (e.g., a thread usually implies the existence of a hole); and
- Functional, or design, features defined by a user, although design features are not necessarily manufacturing features.

Figure 1.

An industrial part (shown from two viewpoints) and its decomposition into four rounded-end slots, two holes, and three open pockets.



¹This computing technique is described in G. Ernst's and A. Newell's *GPS: A Case Study in Generality and Problem Solving*, Academic Press, New York, 1969.

Such evidences contribute to *hints* that specific machining features may exist in a part. For example, in Figure 2, two hole hints were generated from two cylindrical surfaces, a slot hint was generated from two opposing parallel planar faces, and a pocket hint was generated from the bottom, or floor, face of the pocket. Note that the pocket floor intersects all the faces present in the other three hints. (Further hints were also generated but did not become valid features.)

From an implementation point of view, a hint is an incomplete frame representation for a feature. The hint contains a feature type, such as hole or slot, some easily extractable parameters, such as a hole diameter, and links to faces in the solid's boundary. Evidences have *heuristic strengths* reflecting our preferences for certain features (e.g., a slot is easier to machine than a pocket) and our beliefs that hints will lead to valid features (e.g., a complete cylindrical face is more likely to become a valid hole than an incomplete cylinder). Evidence strengths are combined through AI methods involving uncertain reasoning [4] so as to produce strengths for hints. Much like vision systems, IF² reasons with incomplete data. Incompleteness arises in vision primarily because of occlusions, whereas in feature recognition, it is due to feature interactions.

Hints are stored in a blackboard structure [1] that also contains input part models—typically a representation of the part's bounding faces and edges plus a model in terms of design features, such as bosses, webs, and holes—as well as representations for the delta volume and its faces, and for the validated, recognized features. The rankings of hints according to their heuristic strengths are stored as a priority queue, which serves as the blackboard's agenda and provides the system's focus of attention. The control mechanism selects the strongest hint from the agenda and attempts to validate it by invoking the *feature completer* [15].

Feature completion is a complicated geometric procedure involving several calls to a solid modeling system (e.g., to perform Boolean operations). In essence, it extends the faces present in a hint and attempts to compute the largest volumetric feature compatible with the

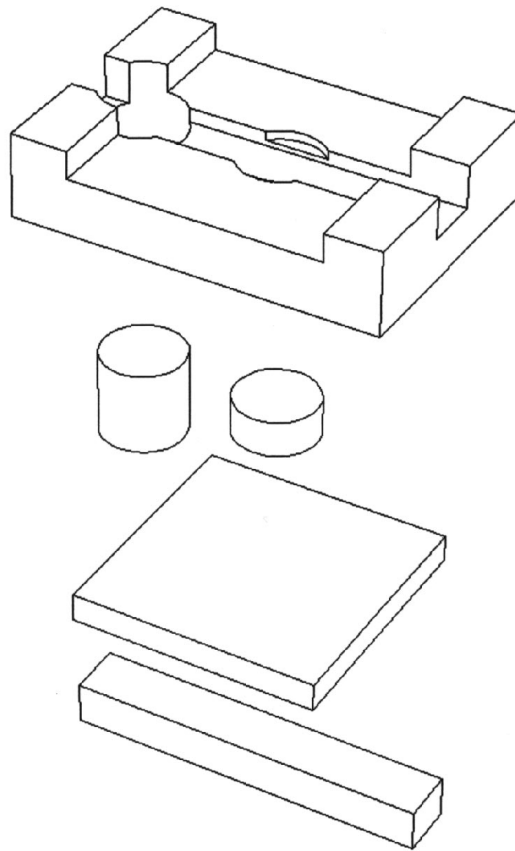


Figure 2.
A test object with four interacting features.

face data while not intruding into the desired part. Accessible directions and other information needed to ensure feature machinability are generated as byproducts of the completion procedure. Figure 2 shows that completion reconstructs the missing portions of faces and edges. If feature completion succeeds, a validated feature is added to the blackboard. A recognized feature provides evidence *against* hints that share faces with the feature because those faces already have an associated feature. Such negative evidences help update the agenda dynamically.

Initially, the material to be removed is the entire delta volume. After each feature is recognized, the removal material is updated by subtracting the feature from it. If the result is null, the process terminates because the delta volume is completely decomposed. The heuristics used to rank the hints ensure the part interpretation generated by the feature finder is desirable and is computed efficiently. However, a downstream module, such as a process planner, may find a feature interpretation unsuitable, possibly forcing an additional clamping and request an alternative. Requesting an alternative can be done easily by deleting the unsuitable feature and retrieving from the blackboard hints that point to the feature's faces.

Some design features (e.g., holes) may correspond directly to manufacturing features, although design systems usually do not guarantee accessibility and other machinability conditions. Other design features, such as bosses, webs, and ribs, are not directly machinable. Our feature finder uses design-feature information by generating strong hints from the design features directly related to machining features. These hints gravitate to the top of the agenda and are usually validated easily [3]. Links between recognized machining features and the design features that contribute

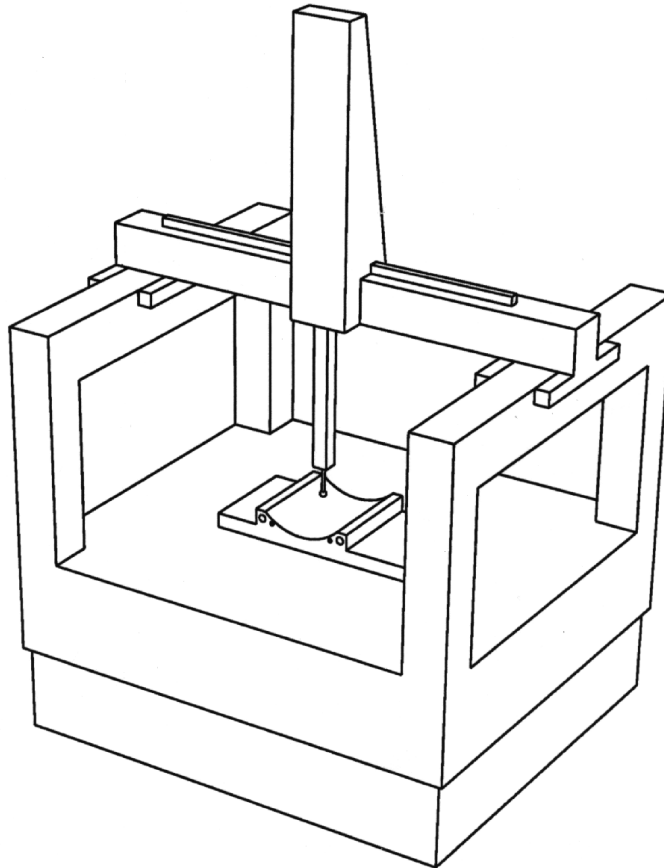


Figure 3.

Output of USC's planner, showing a coordinate measuring machines with its probe in contact with a workpiece.

to them are stored in a simple justification-based truth maintenance system (JTMS) [2]. The recognizer works incrementally, as the design evolves through editing and incremental construction of part models. The machining-feature model is updated efficiently with the help of the JTMS.

IF² is operational, except for its incremental processing capabilities, which are still being implemented. It has been interfaced to a process planner to demonstrate the automatic generation of process plans from solid modeling data and is being interfaced to a feature-based design system.

Planning for Dimensional Inspection

The manufacturing market

increasingly demands high-quality products. Ensuring the quality of mechanical parts and assemblies involves measuring them to verify that their dimensions are within designer-specified tolerances. Dimensional inspection is also crucial in manufacturing process control. Dimensional variations can be correlated with process parameters, and these parameters are adjusted to produce the desired parts.

Coordinate measuring machines (CMMs) are versatile and well suited for integration in automated manufacturing systems. They can provide all necessary measurement capabilities for most parts. A CMM is essentially a very accurate Cartesian robot, equipped with a touch probe for returning the spatial coordinates of the points where the probe contacts a workpiece. Functionally, it is a three-dimensional digitizer. Figure 3 shows a frame of a simulation of a CMM inspecting a mechanical part placed on the CMM table.

For the last few years, we have been developing a high-

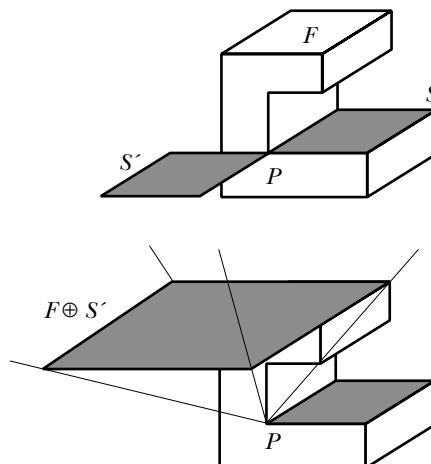


Figure 4.

(Bottom) Steps in the computation of a global accessibility cone: reflecting the feature S about point P , adding S' to a face F , and computing a cone with apex P and cross section $F \oplus S'$.

level dimensional inspection planner, dubbed DIP [11, 12, 13]. The planner computes orientations for setting up the workpiece and determines for each setup the features—typically part faces—to be inspected, as well as the associated probes and probe directions. DIP reasons in a space of evolving plans through operators that map plans to other more refined plans. The state of the planner is represented by 4-tuples of the form (SU, SF, PG, PD), where SU is the setup orientation, SF is the surface feature to be inspected, PG is the probe geometry (selected from a set of available probes), and PD is a set of probe directions, or a *direction cone*.

The initial state includes every direction, feature, and probe geometry and is represented by the 4-tuple (all, all, all, all). This set of plans contains many infeasible and unduly expensive plans. It must be successively constrained by operators that prune out poor plans and seek to reduce the cost of plan execution by avoiding multiple setups, multiple probes, and multiple probe directions. Reorienting a part or probe or changing a probe is a time-consuming operation adversely affecting the accuracy of the results and therefore must be used sparingly.

The most interesting planning operators are those that restrict probe directions to ensure accessibility, and that cluster directions to reduce the number of reorienting maneuvers. Here, we discuss briefly the accessibility operators. Consider the example in Figure 4. The Global Accessibility Cone (GAC) for face S is the set of all directions d that satisfy the following condition: sweeping a half line l of direction d over S with the endpoint of l in contact with every point of S does not cause any intersections of the workpiece with the interior of the line. This definition is the equivalent of saying that a semi-infinite probe of direction d can inspect, or touch, every point of S without colliding with the workpiece. The algorithm for computing GACs may be outlined as follows [12]:

- Reflect the surface feature S to be inspected about point P , yielding S' .
- Compute the Minkowski sum of S' with each face, edge, and vertex of the workpiece.
- For each of these Minkowski sums, compute the cone with apex P and cross-section equal to the Minkowski sum.
- Union all the cones.
- Complement the result.

The second step needs explanation. The Minkowski sum of faces F and S' is shown at the bottom of Figure 4. Note that this sum is simply the result of sweeping one face over the other. It can be shown that only certain easy-to-compute faces, edges, and vertices of the solid contribute to the result; therefore, we need not consider all the boundary elements

of the solid in the second step. The algorithm is relatively simple because it turns out that we need only compute two-dimensional Minkowski sums and that the Boolean operations on cones are essentially two-dimensional because the cones can be represented as subsets of the unit sphere.

DIP is currently being reimplemented in SOAR [5] to provide us with a flexible environment for experimenting with several planning strategies. It is also being extended to compute probe paths by using new path-planning algorithms that exploit the results of accessibility analysis. The ultimate goal is the automatic generation of complete CMM programs.

Conclusions

Manufacturing automation is an interdisciplinary field. Domain knowledge comes primarily from mechanical and industrial engineering, but the tools required to solve the problems are mainly from computer science. Our experience indicates that computer scientists must be willing to get their hands dirty and delve into the domain to work effectively in automation. They must be able to extract and formulate the problems to be solved and understand the domain constraints.


W

e showed here that AI coupled with geometric computation can tackle difficult planning problems in manufacturing and inspection. These

results contribute to the field of intelligent manufacturing, but they are also interesting from the computer science point of view, because AI per se has not had much success in the past dealing with spatial problems.

Major issues requiring further study include efficiency issues (scaling up); more complex geometric domains, such as sculptured or doubly curved objects; additional manufacturing processes, such as casting or stereolithography; and other lifecycle activities, such as maintenance.

Acknowledgments

The research reported in this article was supported in part by the National Science Foundation under grants DMC-87-96192, DDM-87-15404, CDR-87-17322, and DMI-92-14996, by the industrial members of the Institute for Manufacturing and Automation Research (IMAR), and by the Industrial Associates of the Programmable Automation Laboratory, Institute for Robotics and Intelligent Systems (IRIS) of the University of Southern California. Work on the integration of feature recognition and design by features is collaborative with the Fraunhofer Institute for Computer Graphics, Darmstadt, Germany. 

References

1. Englemore, R., and Morgan, T. *Blackboard Systems*. Addison-Wesley, Reading, Mass., 1989.
2. Han, J.-H., and Requicha, A.A.G. Incremental recognition of machining features. In *Proc. of the ASME International Conference on Computers in Engineering*, vol. 1 (Minneapolis, Sept. 11–14), 1994, pp. 143–149.
3. Han, J.-H., and Requicha, A.A.G. Integration of feature based design and feature recognition. In *Proc. of the ASME International Conference on Computers in Engineering* (Boston, Sept. 17–21), 1995, pp. 569–578.
4. Kanal, L.N. and Lemmer, J.F. *Uncertainty in Artificial Intelligence*. North-Holland, Amsterdam, 1986.
5. Laird, J.E., Newell, A., and Rosenbloom, P. S. Soar: An architecture for general intelligence. *Artif. Intell.* 33, 1 (1987), 1–64.
6. Latombe, J.C. *Robot Motion Planning*. Kluwer, Boston, 1991.
7. Requicha, A.A.G. Solid modeling: A 1988 update. In *CAD Based Programming for Sensory Robots*, B. Ravani, Ed. Springer-Verlag, New York, 1988, pp. 3–22.
8. Requicha, A.A.G., and Rossignac, J.R. Solid modeling and beyond. *IEEE Comput. Gr. Appl.* 12, 5 (1992), 31–44.
9. Requicha, A.A.G., and Voelcker, H.B. Solid modeling: A historical summary and contemporary assessment. *IEEE Comput. Gr. Appl.* 2, 2 (Mar. 1982) 9–24.
10. Requicha, A.A.G., and Voelcker, H.B. Solid modeling: Current status and research directions. *IEEE Comp. Gr. Appl.* 3, 7 (Oct. 1983) 25–37.
11. Spyridi, A.J., and Requicha, A.A.G. Accessibility analysis for the automatic inspection of mechanical parts by coordinate measuring machines. In *Proc. of the IEEE International Conference on Robotics & Automation* (Cincinnati, Ohio, May 13–18), 1990, pp. 1284–1289.
12. Spyridi, A.J., and Requicha, A.A.G. Accessibility analysis for polyhedral objects. In *Engineering Systems with Intelligence: Concepts, Tools and Applications*, S.G. Tzafestas, Ed. Kluwer, Dordrecht, Holland, 1991, pp. 317–324.
13. Spyridi, A.J., and Requicha, A.A.G. Automatic programming of coordinate measuring machines. In *Proc. of the IEEE International Conference on Robotics & Automation* (San Diego, Calif., May 8–13), 1994, pp. 1107–1112.
14. Vandenbrande, J.H., and Requicha, A.A.G. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Trans. Pattern Anal. Mach. Intell.* 15, 10 (Dec. 1993) 1269–1285.
15. Vandenbrande, J.H., and Requicha, A.A.G. Geometric computation for the recognition of spatially interacting machining features. In *Advances in Feature-Based Manufacturing*, J.J. Shah, D. Nau, and M. Mantyla, Eds. Elsevier/North Holland, Amsterdam, 1994, pp. 83–106.

About the Author:

ARISTIDES A.G. REQUICHA is a professor of computer science and electrical engineering at the University of Southern California where he directs the Programmable Automation Laboratory and the Laboratory for Molecular Robotics. **Author's Present Address:** Computer Science Department, University of Southern California, 941 West 37th Place, Los Angeles, CA 90089-0781; email: requicha@lipari.usc.edu

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
