SHAPE SYNTHESIS FROM SPARSE, FEATURE-BASED INPUT

by

Steve R. Burgett, Roger T. Bush, S. Shankar Sastry, and Carlo H. Séquin

Memorandum No. UCB/ERL M96/2

29 January 1996

SHAPE SYNTHESIS FROM SPARSE, FEATURE-BASED INPUT

by

Steve R. Burgett, Roger T. Bush, S. Shankar Sastry, and Carlo H. Séquin

Memorandum No. UCB/ERL M96/2

29 January 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Shape Synthesis from Sparse, Feature-Based Input

Steve R. Burgett **Roger T. Bush**
S. Shankar Sastry **Carlo H. Séquin**

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, California USA

Abstract

We are researching a new paradigm for CAD which aims to support the early stages of mechanical design well enough that designers are motivated to use the workstation as a conceptual design tool. At the heart of our approach is *shape synthesis*, the computer generation of part designs. The need for such design automation arises from the fact that many mechanical parts can be defined by two kinds of geometry: features that are critical to its function (*application features*), and the material that merely fleshes out the rest of the part (*bulk shape*). Application features are most often associated with contact surfaces of the part, for example, a bore for a bearing or a mounting surface for a motor. They are the high-level entities in terms of which the designer reasons about the design. Bulk shape must obey certain constraints, such as noninterference with other parts, minimum allowable thickness of the part, etc., but is somewhat arbitrary.

We are developing a system wherein the designer inputs the application features, along with topological constraints, degrees of freedom, and boundary volumes, then the bulk shapes of the parts are synthesized automatically. Overall economy is enhanced by reducing the amount of input necessary from the designer, by providing for more complete exploration of the design space, and by enhancing manufacturability and assemblability of the component parts. This paper presents the functional requirements of such a system, and discusses preliminary results.

Keywords: Concurrent Design, Shape Synthesis, Computer Aided Design, Design Automation.

1 Introduction

Today's mechanical CAD programs require a user to specify a design using mostly explicit geometry. However, a CAD application that relies on explicit geometric input is not well matched to the design process and thus is usually relegated to the task of design documentation rather than a true aid to designers [1]. In the early stages of design, only a fraction of the geometric elements are known, yet solid modelling software requires models to have physically realizable shapes, even for simple visualization. At any stage of design, routine changes such as resolving interference between two parts may require extensive alteration to the geometric database—a laborious process for the designer.

Not only is it time consuming to edit explicit geometry, it has been noted that geometry alone is incomplete as a design specification language [2]. After a design is complete, much valuable information is not represented. It is difficult to decide what can be changed, what parts of the design are crucial, and why certain decisions were made in the first place. In fact, a major impetus for *feature recognition* research is the inference of knowledge lost during the design process [3]. A method of specification that facilitates the design process and is richer in information is necessary as a foundation for future mechanical CAD systems. Further, if it is to be useful, this method of specification must have a way of automatically resolving itself into a consistent design that is expressed in explicit geometry.

Approaches to better specification paradigms can be roughly grouped into geometry-based and artificial intelligence (AI) based. The geometry-based approaches combine pieces of geometry into groups and augment them with higher level information. These are generally classed as *Feature Based Design* methods. The term *feature* is subject to a variety of interpretations. In its broadest definition, "a feature is an element used in generating, analyzing, or evaluating a design."[3] Design-with-features approaches are similar to solid modellers in the level of automation they provide. They still require that parts be explicitly defined by a combination of features—the emphasis is on recording more information than just geometry. In contrast, AI-based paradigms usually seek to provide automation by reasoning in terms of higher-level entities like machine components. They include constraint satisfaction systems and knowledge-based systems. These methods tend to be domain dependent (e.g. for designing helical gears), and do not handle the geometric design well, if at all [4]. For example, an AI system might determine the required size of a bearing, or select an appropriate motor, but rarely is consideration given to designing appropriately shaped parts to hold that bearing or motor in place.

A new approach that holds promise is based on the realization that any mechanical part is defined by two kinds of geometry: features that are critical to its function (*application features*), and the material that merely fleshes out the rest of the part (*bulk shape*). Several authors have made this observation [1, 4, 5]. Shapiro and Voelker [6] introduce a new view to interpret mechanical function in terms of energy exchanges and to consider some key portions of geometry as *energy ports.* These are the discrete subsets of a system's physical boundary through which it interacts with its environment. Energy ports on a mechanical part exist at contact surfaces, such as the internal surface of a bearing bore. Their shape is usually dictated by the nature of the contact (e.g. once a specific bearing has been selected, the dimensions of the bore are implied). Thus, the geometries of the energy ports are fully specified, and can be considered to be the *application features* of the part to be designed. In other words, an application feature is a *known geometric solution to a local problem.*

In a typical design cycle, the geometries of the application features are known by the designer before that of the bulk shape. **A missing link in current approaches to design automation is the automatic synthesis of bulk geometry**. Since a part's function is determined by its application features, it is possible to automate the synthesis of the bulk shape, using the application features as a design specification. A few authors have begun to explore *shape synthesis* on these terms. Duffey and Dixon [7] automate the design of cross sections for extruded beams given load and support points and forbidden areas. Graham and Ulrich [8] automate the design of 2-D bending patterns for sheet metal parts using path planning and iterative refinement. Given application features and paths, Shimada and Gossard [9] generate skinning boundaries by solving for the shape of a deformable curve in a potential energy field. **In this paper we present a paradigm for integrating shape synthesis into the design environment.**

In the system we are designing, the interface is a feature-based design environment wherein the user manipulates application features directly. All reasoning about the design is provided by the user. Synthesis of bulk shapes is provided by a *shape synthesis engine.* Alternatively, one could imagine decoupling the shape synthesis engine from the user interface for use as an *agent* in a system of collaborating software agents. In that case reasoning about the design may be provided by other agents.

In our system, user input takes the form of interactively specified application features, design constraints, manufacturing process information, prefabricated vendor parts, and ranges of motion. Higher-level information, such as the kinds associated with design-with features, is not required by the shape synthesis engine. It requires only that a geometric representation be extractable from each feature, that each feature's membership in a specific part be indicated, and that the relative motions of those parts be available. **Part geometry need not be completely specified by the user.** Much of the geometry of the design is then synthesized automatically by the program.

In contrast to currently available CAD modellers, this approach will strongly support the early stages of design (conceptual design), where widely varying concepts are explored. There are several advantages of such a system over a conventional solid modeller:

- Far less tedium is imposed on the designer because the need for explicit detailed geometric specification is greatly reduced.
- Rapid design space exploration is facilitated and thus can be much more thorough.
- Manufacturing process knowledge can be applied at the part synthesis stage to yield parts that are cheaper and faster to manufacture.
- Design of the assembly sequence can influence the shape synthesis process and vice versa.
- The design is "live." If a subdesign is dropped into a larger context or assembly, interferences and conflicts can be automatically resolved.
- A measure of design intent is implicitly encoded. Less documentation and explanation is needed to pass a design from one designer to another because the design is represented in a more functional way.

1.1 A Design Example

To illustrate the utility of this application of shape synthesis, we describe a design example. Suppose a design specification requires that a tool have a linear motion along a given axis over a given range (see figure 1). This is the only design requirement; any other parts introduced will exist solely to support this single function. The designer knows that there must be a linear bearing with its axis parallel to the tool's. Further, she knows that certain connectivity relationships exist: one side of the bearing must be rigidly coupled to the tool, the other side to the machine baseplate. These relationships imply the existence of *glue* parts (shown in dashed lines in

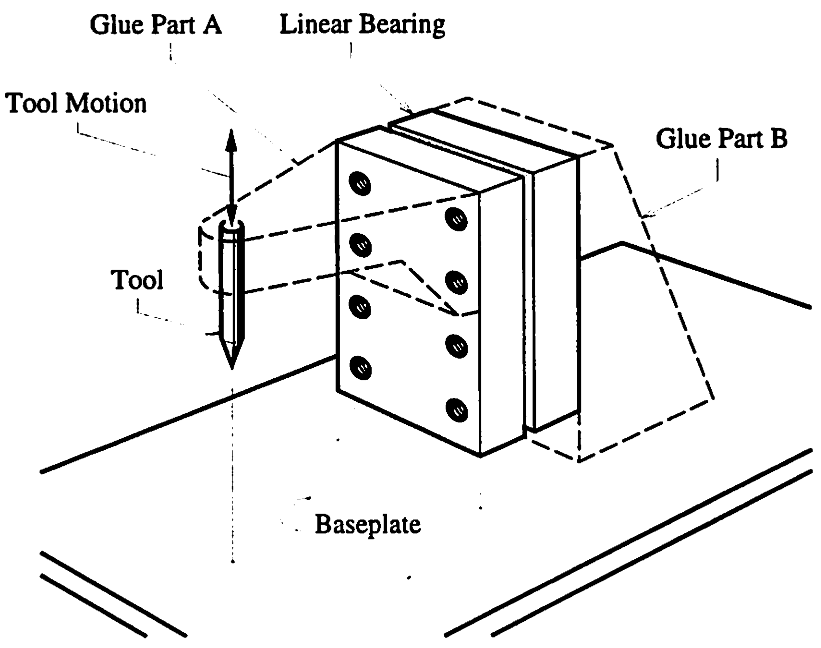

Figure 1: A design example.

the figure), whose sole function is to hold the primary parts fixed in proper orientation to each other.

In the design system we envision, the designer would proceed as follows: Since she knows there must be a linear bearing, she selects one from an on-line catalog of off-the-shelf parts. She then interactively positions it in three-dimensional space relative to the baseplate, and indicates that a part must connect the two. At this point a glue part is automatically synthesized (B in the figure) and displayed showing the bearing mounted to the baseplate. In addition, the synthesized part has the appropriate mating holes and doesn't interfere with the linear bearing throughout the range of its motion. Further, it is manufacturable by the currently selected process, say milling with a three-axis, numerically controlled milling machine. If she decides that the part should be a little more rugged, she increases the minimum allowable thickness for the part and it is automatically resynthesized.

The designer next chooses a tool from the on-line catalog and positions it in space. She selects the linear bearing and the tool to be connected and a part is automatically synthesized that mounts the two (A in the figure). Finding that the tool must be repositioned, the designer drags it to the

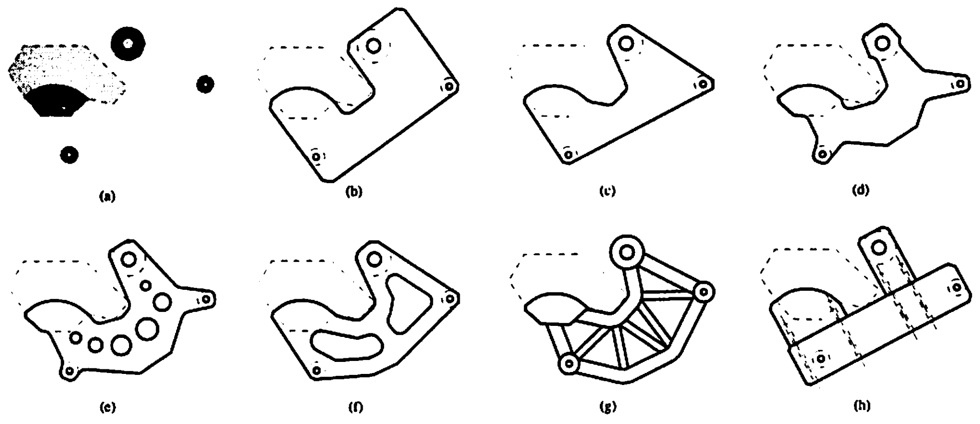

Figure 2: A specification and seven conforming designs.

correct place. As she does so the glue part is resynthesized to accommodate the new position and avoid interferences.

With this design stage completed, this subassembly can be imported into a higher-level assembly. As it is positioned in the machine, there may be several points of interference that must be resolved. Synthesized portions of parts in the assembly and subassembly can be automatically adjusted to eliminate many of these, minimizing the burden on the designer.

1.2 Observations

Our enthusiasm for this new paradigm is based on several observations:

- During conceptual design exploration, only the application features are interesting, e.g. bearing bores, bolt patterns, and load contact points.
- The bulk shape of a part determines much about the manufacture of the part and vice versa.
- Every part design starts with an initial guess. The refinement of this design depends on the relative importance of manufacturing costs, part performance, etc.

The first observation, as Shapiro and Voelker [6] point out, is that in a typical design cycle the geometries of the application features are known by the designer before that of the bulk shape. In other words, during conceptual design exploration, only certain details of each part are interesting—the *application features*. The rest of the shape of the part, the *bulk shape*, is much more arbitrary. It must obey geometric constraints such as connectivity and noninterference with other parts, but this allows an infinite variety of actual shapes.

Consider the example in figure 2, which shows specification for a 2-D mechanical part and several conforming designs. In this example, a part is needed with several holes and a curved edge (the application features), and there is an *illegal region* that it must avoid. Strength requirements dictate a minimum thickness of material around each hole and behind the curved edge (dark shading). The curved edge contacts some object during operation, which gives rise to the illegal region (light shading). There are infinitely many part designs that satisfy this specification, and seven such solutions are shown. They all share the same application features, yet differ in their bulk shapes. (See also figure 6.)

A second observation is that the bulk shape of a part determines much about the manufacture of the part and vice versa. In many cases, a designer will choose the bulk shape of a part based on knowledge of manufacturing considerations. A shortcoming of using a conventional solid modeller (or even sketching on the back of an envelope) is that since the designer is forced to give *some* shape to each part, she introduces a bias about how that part is likely to be manufactured long before it is desirable to do so. For example, if a part is initially modelled as though it will be milled, it becomes difficult to alter the CAD model to show the part as it would be if stamped and bent from sheet metal.

Another fundamental manufacturing issue is the choice of stock from which the part will be made. There might be an alternative between hogging the part out of a single block versus welding it from bars and plates. Even the simple flat-plate part in figure 2 can be manufactured in many different ways. Design (b) could be made easily on a three-axis milling machine with mostly x and y cuts. If the scale of the problem is larger, design (h) would make more efficient use of stock. Designs (e) and (f) might be appropriate for stamping or if weight is to be minimized. At very large scales, design (g) could be made from bent and welded bars or box-section beam. Ideally, the designer would be able to browse through various manufacturing options at any point in the design process, thus freeing the conceptual phase from such concerns. Conventional CAD systems offer no support for exploring the design variations necessary to accommodate this.

Our final observation is that all part designs require an initial guess. Depending upon the domain, this guess will get refined in different ways. A component in the landing gear of an aircraft, for example, will be subject to iterations of finite element analysis and shape revisions until a satisfactory strength to weight ratio is achieved. In other domains manufacturing and assembly concerns dominate. In high-volume products, only the most cost-efficient processes are practical, like injection molding and sheet metal stamping. The nature of the chosen process will dictate the bulk shape of the parts.

On the other hand, a large class of design problems are best satisfied by parts with simple shapes because these are quick to manufacture with processes like milling. Examples include prototypes, custom laboratory equipment, tooling, jigs, fixtures and even low-volume for-market products. Such parts are traditionally designed by intuition and rules of thumb. Mechanical requirements, like strength and stiffness, are met simply by oversizing the parts by a large margin. The primary consideration for such parts is the ability to quickly design and manufacture them.

In all of these cases the bulk shape of the part must obey geometric constraints. In the latter cases, design criteria such as ease of milling can be translated to geometric constraints as well, such that satisfactory shapes could be generated automatically. In the case of more highly engineered parts, like aircraft parts, the synthesizer can generate a sensible initial guess for input to the refinement process.

1.3 Design System

Motivated by these observations, we are developing a system wherein the designer enters the application features of each part into the system, then the software rapidly synthesizes bulk shapes to support these features. The synthesis will observe implicit geometric constraints such as connectivity and noninterference. Considerations of strength, manufacturing methods, and so forth, will also control the synthesizer. In a sense, the functional specification of the design is captured in the feature system that is input by the designer. By changing parameters to the shape synthesizer, the designer can examine various solutions that satisfy the functional specification.

This description could equally well be applied to the system developed by Shimada and Gossard [9]. Our goal is to expand on those ideas to present the user with a selection of shape styles. A basic tenet of our research is that the CAD system and the user should work in concert, and the link between them should be a highly interactive interface. Further, we strive to minimize the volume of information required from the user, particularly during conceptual design. For example, we do not want to require the designer to specify all of the loads that will be applied to each part. In many cases this would be inappropriate. Designers often don't know what the loads will be beyond the points of application and a general feeling of "large" or "small." Further, it is often desirable to design a part to withstand not just service loads, but general loads from all directions, to make it robust to unmodelled loads, assembly, and accident. This is especially true when designing prototypes, tooling, and other applications where parts are traditionally designed "by eye." Our approach will be to synthesize designs that look reasonable, and present them to the designer for assessment. If the part looks too weak, she may choose a different style, or may issue commands like "fatten it up here," until the part is satisfactory. In other words, the system we envision is not so much one of design automation, but one of designer facilitation.

Finally, since our application will decide on the specific part geometry, manufacturing process knowledge can be applied at the part synthesis stage to yield parts that are cheap and easy to manufacture. If geometric operations that correspond to actual machine steps are used (e.g. drill hole, angled cut), a construction plan is implicitly generated with the part. This simulates the destructive solid geometry approach [10] without requiring the designer to work in terms of these low-level primitives. Global part properties, such as manufacturing process or style of construction, may be changed with relative ease because the shape synthesizer is responsible for the details of the geometry and the manufacturing plan.

2 Shape Synthesis

2.1 Part Shapes

The question that must be addressed is, "what should synthesized designs look like?" To answer this, we begin by categorizing the kinds of requirements that mechanical parts must meet. Though not comprehensive, the following list includes the most common and most important categories:

1. Connectivity: A part must be a single, connected, physical body.

2. Interference: The part must not interfere with any other parts in the machine, their paths of motion, nor with illegal regions arising in any other way.

3. Strength and Stiffness: The part must be strong enough not to fail in service or during assembly, and it must be stiff enough to perform its function correctly.

4. Manufacturability and Assemblability: The part design must be able to be fabricated by cost-effective means, and the machine itself must be able to be assembled efficiently.

5. Dynamics: A moving part must meet acceptable ranges for mass, moments of inertia, resonant modes, etc., as defined by its function.

6. Electromagnetic characteristics, aerodynamics, hydrodynamics, aesthetic styling, thermal expansion, etc.

This list is shown in an approximate order. Those requirements near the top must be met by *all* parts, regardless of the application. As we move down the list, the requirements become progressively more specialized, and may or may not be important for a given part.

2.2 Division of Labor

We have chosen to divide the synthesis problem into two steps. In the first step we will generate a *skeleton* which abstracts the structure of the part. That is followed by a *material synthesis* step which fleshes out the actual shape of the part using the skeleton as a guide. This is identical to the procedure of Shimada and Gossard [9], who apply the term skeleton to the energy paths of Shapiro and Voelker [6]. We also observe that the skeleton is analogous to the STICKS diagram [11] used in CAD for integrated circuit design.

There are alternatives to this two step approach. For example, Duffey and Dixon [7] use recursive application of topological operators to generate designs for extrusions. In figures 2 and 6, there are examples which could be synthesized in a single step. Design 2(b) is a modification of a rectilinear bounding box. Designs 2(c) and 6(b) are modifications of the convex hull of the features. In these cases the modification is essentially the removal of illegal regions.

In more complex cases however, these simple design styles will need some help. For example, if an illegal region completely divides the bounding box or convex hull, it becomes unclear where material must be added to make a connected part. Further, the more refined design styles are much easier to generate if we first abstract their structure with a skeleton. Designs 2(d), (e), (h), and 6(d), (g), and (h) are based on tree-type skeletons. Designs 2(f), (g) and 6(b), (e), and (f) are based on triangulations.

The skeleton also has the advantage of being an intuitive intermediate abstraction. If the designer feels the need to make minor modifications to the synthesized skeleton, it would be natural to add or delete paths to change its overall character—a process much faster than editing explicit geometry. In this way, the skeleton can be seen as a tool to manage the vast complexity of the design space. It is a representation of an infinite family of parts with different geometry, but the same topology.

Skeletons (which have zero width), are synthesized so that they can be widened by some amount without interfering with illegal regions. This amount will be dictated by the current design rules. The first action of the material synthesizer is to perform this widening, yielding a *thick skeleton.* This thick skeleton is essentially the minimally sufficient part, satisfying the required connectivity and providing at least the minimum allowable material thickness everywhere. The remainder of material synthesis adds material to the thick skeleton to improve manufacturability, reworkability, and assemblability. Figure 6(a) and (c) depicts two styles of thick skeleton for the same specification.

A further advantage of separating the synthesis process into two steps is that it allows us to mix and match skeleton and material synthesis algorithms and thus multiply the number of design styles we are capable of generating. Some combinations may not be as useful as others, of course, but we should still realize a rich selection.

3 Program Structure

Figure 3 depicts the program flow schematically. A strength of this paradigm is that it separates into fairly independent components, and it allows a single-pass program flow (complications are discussed in Section 3.6). In the diagram all information flows down. The user inputs a functional description of the machine, which includes descriptions of the application features of each part and its position or relative motion within the machine. There may also be stationary or moving illegal regions that the designer wishes to reserve so that no material will be synthesized in them. This information is preprocessed to yield a set of required and illegal regions. The set is then passed to the shape synthesizer, which consists of the skeleton and material synthesizers.

3.1 Input

As stated above, the input to our system includes application features, part positions and motions, and reserved illegal regions. In an ideal implementation the user would have some help from an online catalog. This would

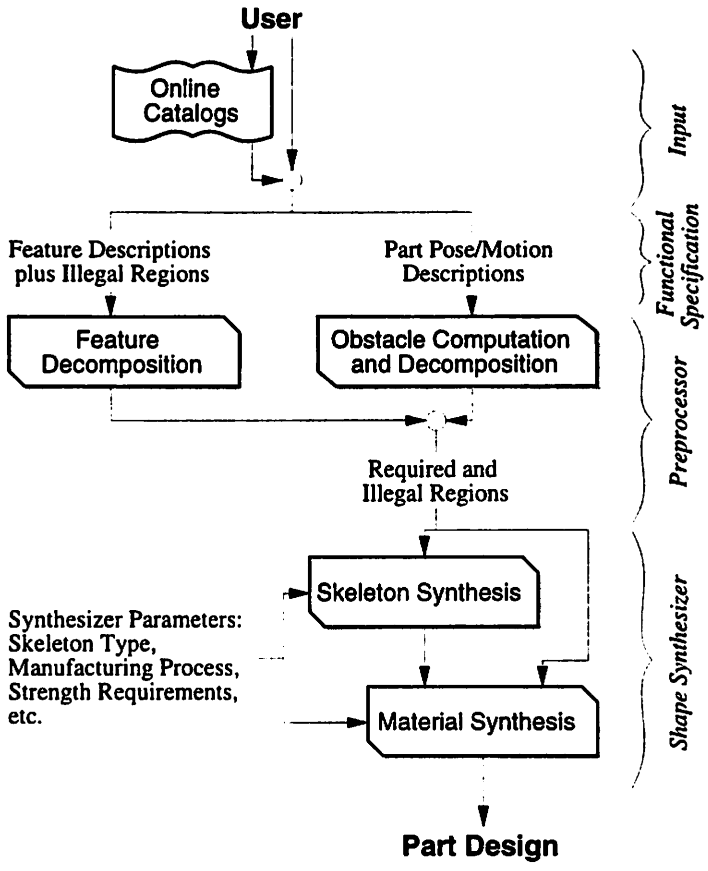
User
Online Catalogs
Feature Descriptions plus Illegal Regions
Part Pose/Motion Descriptions
Feature Decomposition
Obstacle Computation and Decomposition
Required and Illegal Regions
Skeleton Synthesis
Synthesizer Parameters: Skeleton Type, Manufacturing Process, Strength Requirements, etc.
Material Synthesis
Part Design
Input
Functional Specification
Preprocessor
Shape Synthesizer

Figure 3: Program Flow.

be a database of off-the-shelf parts that can be included in the design of the machine. In addition to modelling the parts themselves, each database entry would contain parametric information about the features that glue parts must have to support it.

Recall the example in figure 1. The shape of glue part A is defined by the mating requirements of the tool and the linear bearing. Specifying this part could be as simple as selecting the tool and bearing from the catalog, and commanding a glue part between them. The program would extract the necessary mounting surfaces, threaded holes, etc., from each model, and synthesize a part that has these features. The addition of "interface geometry" to part databases has been proposed by Johannesson [12].

In figure 3, the information yielded by the input process includes applica-

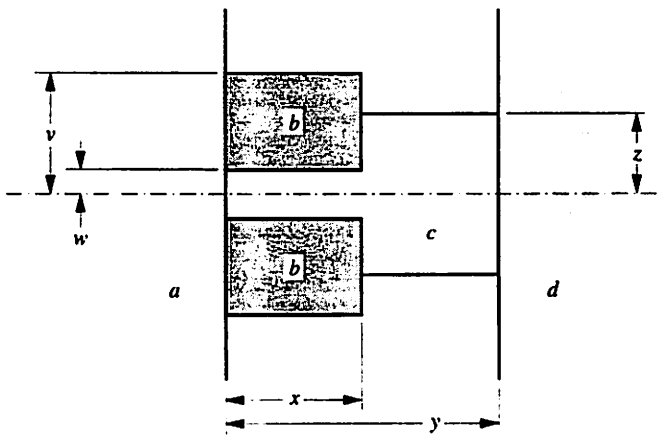

Figure 4: A feature is composed of required and illegal regions.

tion feature descriptions for each part, general illegal regions, part positions, and motion descriptions. This forms a kind of functional specification of the design.

3.2 Feature Decomposition

The shape synthesizer needs its input in the form of required and illegal regions, connectivity relationships, and design rules. Thus, features must somehow be expressed in terms of these more primitive entities. In practice, we will view a feature as a sort of macro that specifies a structure of these entities. Let us use the term *target part* to refer to the one currently being synthesized. Some illegal regions apply only to the target part, others apply only to other parts, and some apply to all parts. Figure 4 illustrates a hypothetical bolt-hole feature in cross section. Region *b* is an annulus of required material, and applies to the target part. Region *a* is an illegal region which arises from the mating part, and applies to all parts in the design except the mating part. Region *c* reserves space for the bolt and applies to all parts except the bolt, washers, nut, etc. Region *d* is an illegal halfspace which reserves access for a tool to insert and tighten the bolt. It may apply to all parts, or only to parts that come earlier in the assembly sequence. To instantiate a bolt hole, the user would supply the dimensions v, w, x, y, and z.

This is a very simple example. If it were undesirable to reserve the entire halfspace *d* so close to the bolt, an additional cylindrical region might be interposed between regions *c* and *d* to reserve space for a socket wrench and extension.

In figure 3 the output from the feature decomposition step is a system of required and illegal regions suitable for input to the shape synthesizer. This system is part-specific, and must be recomputed for each part to be synthesized. The system must be computed in the frame of reference of the target part, and include only illegal regions applicable to that part.

3.3 Illegal Region Computation

As described above, illegal regions are generated by feature decomposition, and may also be input explicitly by the user. In addition, illegal regions arise because, from the point of view of the target part, all other parts define illegal regions. By extension, if a part moves relative to the target part, the entire space it sweeps out, as viewed from the coordinate frame of the target part, is an illegal region. Note that our whole approach of starting with illegal regions and synthesizing noninterfering parts contrasts with the method supported by traditional CAD systems. There the designer models all of the parts, then has the software check for interference. If any is found, she edits the offending geometry.

In our system, all applicable illegal regions must be computed so that they can be used as input to the shape synthesizer. In the case of parts that are stationary relative to the target part, this is trivial. For parts that move relative to the target part, we must do some work. Though it may at first appear difficult to synthesize shapes in the presence of moving obstacles, it turns out that the problem of synthesis is separable from the motion if we make one assumption: we must have full descriptions of the relative motions of all parts in the machine. In many cases, this will be easy for the designer to input, because the motion of the various parts is intimately related to their application features. For example, a bearing bore is associated with a revolute degree of freedom, coaxial with the bore. Most moving parts have simple kinematics—either a revolute or prismatic axis of movement relative to some other part. Even complex motions, such as those of linkages, can be simple for the designer to input by specifying how parts are connected with joints. Many of the degrees of freedom in a machine will be associated with off-the-shelf parts like bearings and crossed-roller slides. Their entries in an online catalog would include information about degrees of freedom they provide.

With this information in hand, we can compute *static* illegal regions for all parts in the machine. This is important: **shape synthesis for the target part can always be done in a static environment.** The illegal

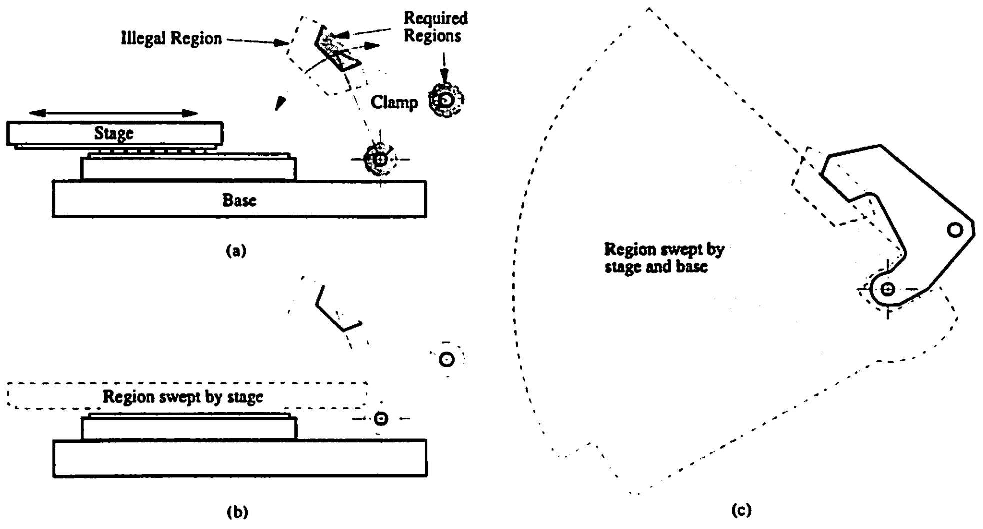

Figure 5: Generating illegal regions by sweeping.

regions that populate this environment are the swept volumes of the other parts in the machine. Each part is swept along its degrees of freedom, as viewed from the frame of reference of the target part. If a part has multiple degrees of freedom relative to the target part, it must be swept recursively, starting with the axis most distal from the target part along the kinematic chain. The process is similar to the computation of the work envelope of a robot [13, 14].

An example is shown in figure 5. At (a) the target part, a clamp, is shown as a set of desired features. It is to have one revolute degree of freedom relative to the base plate, with joint limits at +45 and −30 degrees. At the left is a stage which has a prismatic degree of freedom relative to the base. The stage and the clamp thus have two degrees of freedom relative to each other. To compute all the illegal regions that apply to the clamp, we first attach ourselves to its frame of reference. Starting with the part farthest down the kinematic chain (the stage), we sweep along its axis through its complete range of motion. This gives rise to an intermediate illegal region, shown (shaded) at (b). Proceeding along the kinematic chain toward the target part, we sweep these intermediate regions, plus other parts, along their degrees of freedom. The final illegal region applicable to this target part is shown at (c), along with a possible design.

In this example, we have assumed that the individual degrees of freedom are fully independent. This is the most conservative assumption, yielding the largest illegal regions, but it may not always be appropriate. There could be mechanical stops, or the actuation could be under software control, such that, say, the clamp only moves when the stage is at one specific position. To handle this we need only modify the sweeping process slightly. Rather than sweeping the intermediate region recursively, the stage would be swept along both axes separately, and the union of the two resulting regions would be used as the illegal region.

Any kind of interdependence between part motions can be handled similarly, as long as we are provided with an accurate description of the rules governing these dependencies. Note that if the dependencies are imposed by software control, it is worthwhile to discard them wherever possible and use the most conservative illegal regions. This guarantees that no software error can ever cause a physical machine crash. We also note that linkages do not represent dependent degrees of freedom between their links, just complex degrees of freedom. Computation of the swept areas of planar links has been studied by Ling and Chase [15].

3.4 Skeleton Synthesis

As mentioned earlier, the heart of our system is shape synthesis, which we divide into skeleton synthesis and material synthesis. We represent a skeleton as a graph plus a Euclidean embedding, where the embedding may have non-straight edges. Algorithms to generate appropriate skeletons are a primary area of research. Graphs include several specializations, such as triangulations and spanning trees, and there are two broad categories of approach: computational geometry and numerical optimization. Of interest to us is a class of definitions that require the graph to meet some cost based on geometric characteristics of the edge set, such as length of edges or length of node-to-node paths. Many such graphs are treated in the literature, including the Euclidean minimum spanning tree, Steiner minimum spanning tree,[16] t-spanners,[17, 18] and Delaunay triangulations [19]. The differences between these are the nature of the cost function that the graph must satisfy, and whether the algorithm can add nodes.

Regardless of the algorithm we use to create the skeleton, we must ensure that it can be widened appropriately to create the thick skeleton without interfering with any illegal regions. In its simplest form, the problem is analogous to finding a path for a mobile robot through a roomful of obstacles, and we can borrow the algorithms used to solve those problems. This technique was investigated by Graham and Ulrich [8]. Most techniques account for the size of the robot by transforming the problem to *configuration space*

(CSPACE) [20, 14]. In the case of a circular robot moving through a two-dimensional field of obstacles, CSPACE is parameterized by the coordinates of the center of the robot. For skeleton synthesis, CSPACE is parameterized by the coordinates of a point on the skeleton. *Obstacles* in CSPACE are sets that must not contain any point on the skeleton, lest the thick skeleton interfere with an illegal region in physical space. If the thick skeleton is to be of uniform width, CSPACE is computed simply by growing the illegal regions by one-half the width of the thick skeleton. The skeleton synthesis algorithm can then operate in this transformed space to plan the skeleton, and we will be assured of being able to grow the thick skeleton without interference. In fact, clearance can be added by growing the obstacles an extra amount. If the thick skeleton is to have varying widths, e.g. tapered struts, CSPACE must acquire an extra dimension to account for the half-width of the skeleton. For a 2-D design problem, CSPACE becomes three-dimensional and obstacles are generalized frustums whose tops are the physical illegal regions, and whose sides have a slope of unity. This is an issue we are currently researching.

In all of the skeletonizing algorithms we will consider, there are two general approaches to avoiding illegal regions. The first implementation generates a connectivity graph without regard for the illegal regions, then uses path planning to route the edges. The second implementation directly generates a skeleton in CSPACE that respects the illegal regions. In most cases this produces a superior skeleton, but many of the algorithms studied in the literature have not been extended to handle obstacles.

We are investigating both computationally exact and numerically approximate methods to compute these graphs. Some heuristics use a hybrid approach, where a starting point is found using computational geometry, then an optimization is performed to yield the final result. Numerical methods will often start by computing some structure for which a good algorithm exists, like the minimum spanning tree. For example, some heuristics for approximating the Steiner minimum spanning tree start this way, then choose sites for additional nodes, and finally drive the graph to a (local) minimum weight by moving these new nodes [16].

3.5 Material Synthesis

The material synthesizer has the responsibility of generating the final shape of the part. As it does so, it follows design rules relating to minimum material width, manufacturing methods, assemblability, etc. The material synthesizer must guarantee that the finished shape completely contains the skeleton, and that no portion violates any design rules or illegal regions. The output of the

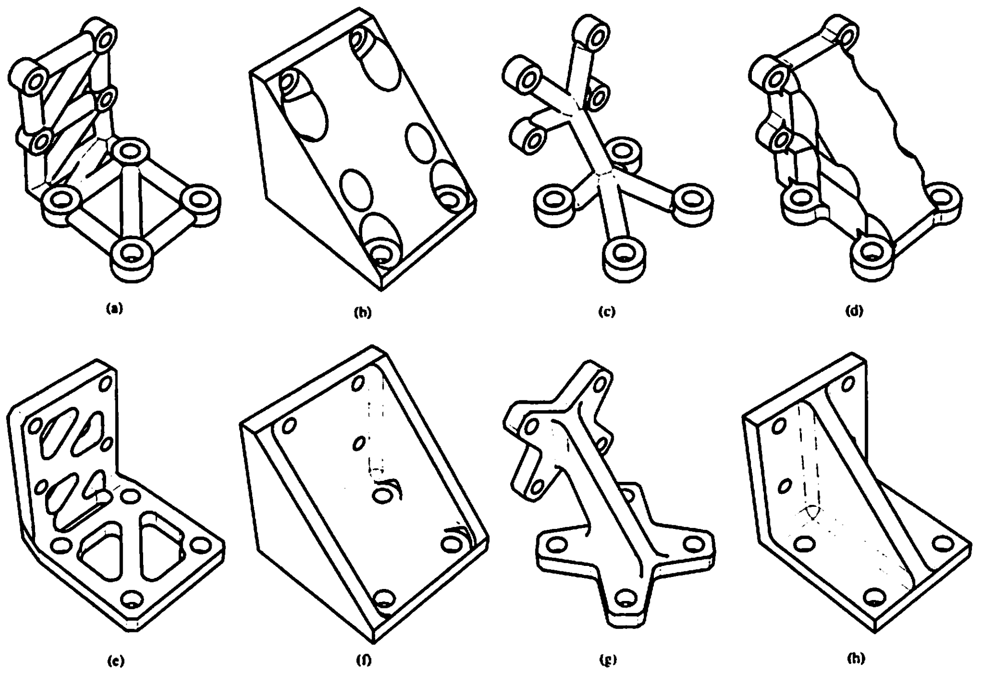

Figure 6: Two thick skeletons (A & C) and six designs, all satisfying the same specification (not shown).

material synthesizer is the actual part design. Shimada's and Gossard's [9] elegant method of generating skinning boundaries by placing a deformable curve in a potential field represents one possible material synthesis style. Our goal is to be able to generate a wide selection of part styles, some similar to those shown in figures 2 and 6. [1]

Of the material synthesis techniques we are considering, the simplest is to approximate the thick skeleton using the selected manufacturing process. For example, if the selected process is milling, material is added to the thick skeleton until the part is composed of flat faces and fillets. Examples are seen in figure 6 (e), (g) and 7. An interesting problem to be solved then is how to synthesize skeletons that ensure sufficient clearance for fillets to be added without creating interferences.

Perhaps the next simplest material synthesis technique is to conceptually wrap a skin around the thick skeleton and pull it tight, letting it drape

[1]It is worth noting that the parts illustrated in fig 6 took over 30 hours to model interactively in Pro/ENGINEER [21]. This highlights the difficulty of using solid modellers to explore design space.

around the illegal regions. Figure 2(c) shows one such design in two dimensions. This can be thought of as a modification of the convex hull of the application features.

More sophisticated material synthesis will add more knowledge about the selected manufacturing process. We envision a planner that reasons in terms of subtractive operations that correspond to milling, turning, drilling, etc., and additive operations that correspond to bolting and welding. This knowledge will be used to add material strategically to yield an easily-manufacturable part. This raises the specter of the well-known difficulties of automatically generating manufacturing plans for a given design. In this case the problem is easier because the synthesizer has the freedom to alter the shape of the part.

In most cases it will be desirable for the material synthesizer to generate axis-aligned geometry wherever possible. This will yield parts that are easier to machine, inspect, and assemble than those with freeform geometry. Figure 2(h) displays an example with almost exclusively rectilinear geometry, typical of built-up parts in a prototype. In figure 6, all six designs exhibit geometry that is aligned to one or more principle axes. Solutions (e) through (h) have all geometry restricted to axis-aligned slabs.

In the case of prototyping, evolution of the design often requires part designs to be modified. When possible, it is often faster and more economical to re-machine the same physical part than to make a new part from scratch. It may be desirable for the synthesizer to add excess material judiciously to improve reworkability of the parts.

3.6 Complications

There are several issues which complicate using the single-pass approach we have outlined. The most important complication is the simultaneous design of multiple parts. In this single-pass model, a problem arises when we must synthesize designs for all the parts in a machine: illegal regions cannot be computed for parts whose shape is not yet known. Our first approach to solving this problem is to use prioritization and iteration. The application features of a part should provide a good placeholder for the space that it will ultimately occupy. We can imagine that the designer first inputs all features of all parts, then to each part she assigns a priority. The synthesizer begins with the highest-priority part, using the application features of all other parts as illegal regions. One by one, each of the parts are synthesized in order of descending priority.

Clearly, prioritization will not always work. It may happen that the i^{th} part cannot be synthesized at all because of the way a higher-priority was synthesized. Sometimes a reordering of the part priorities will solve this, but not always. In some cases, each part must give a little so that both can coexist. Another approach is to try to carve up space and apportion a region to each part before any synthesis begins. A Voronoi partitioning based on application features is one possibility. In this scheme space is divided into cells, each belonging to a particular part. The shape of each cell is determined so that at any point inside the cell, the nearest application feature is one belonging to the same part as the cell. In other words, the cell walls appear "halfway" between the nearest features of adjacent parts. This may be modified to a *weighted* Voronoi partition if it is necessary to give some parts proportionally more space.

A further complication is that the two stages of shape synthesis may have to become more tightly integrated than we have described. For example, the material synthesizer may be able to recognize potential improvements to the design that require modification of the skeleton it was passed. For instance, it might be that a simpler machining operation would be usable if some portion of the skeleton were moved a bit. Iteration between the skeleton and material synthesizers is a possible approach to try first, and can be viewed as a form of optimization.

Finally, it remains to be seen how natural it will be for designers to input designs as systems of features and illegal regions. For small models it seems straightforward, but it may require considerable imagination on the part of the designer for complex systems. We are hopeful that rapid shape synthesis will come to the rescue. If a designer is able to see representative solid bodies for the parts, it should greatly facilitate her understanding of the design she is constructing. Finding an answer to this question is one of the main motivations for implementing our system.

4 Work In Progress

We have developed software that minimally demonstrates a design environment with integrated shape synthesis. It allows the design of a single flat part (2-D) through interactive editing of features, illegal regions, and global part properties. Features are specified by selecting a feature type and positioning it in the design with the mouse. Illegal regions, represented by polygons, are specified and modified interactively. A thick skeleton is generated that

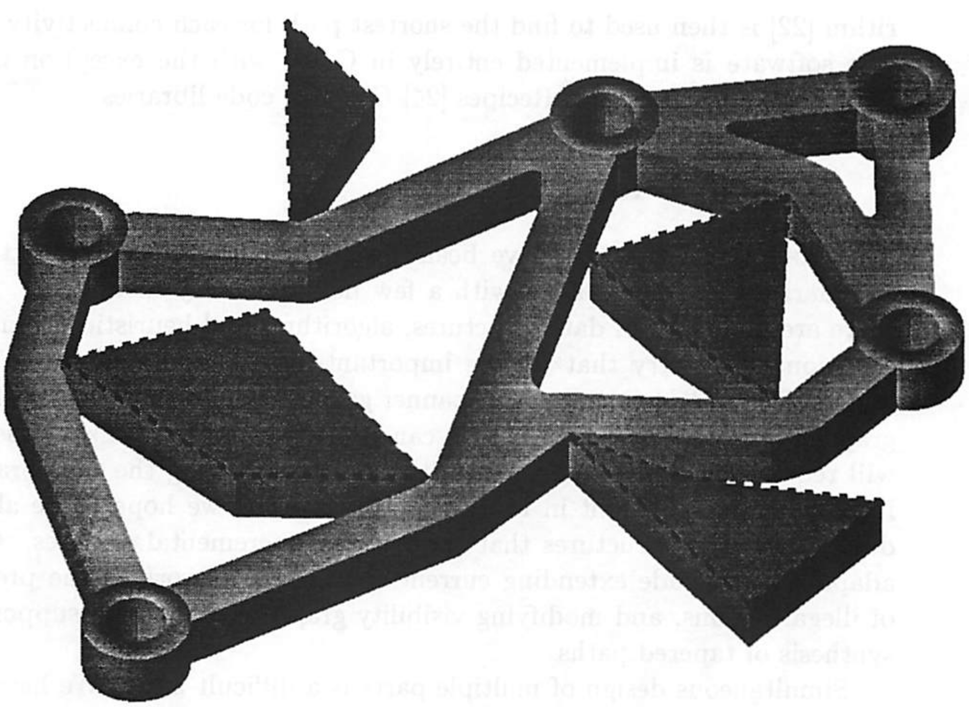

Figure 7: A 2-D part synthesized by the prototype software.

connects the features and avoids the illegal regions. The resulting design is rendered by extruding the part into the third dimension, and resembles a physical part that could be made by bending and welding bars, or by milling. An example is shown in figure 7.

In addition to specifying local features (application features) we support global attributes for parts. Two such part attributes are spar width and illegal region clearance. The user can modify these attributes for the part via an on-screen slider, and see the part dynamically resynthesize. The algorithms are currently fast enough to resynthesize a reasonably complex part several times per second, thus when the user interactively drags a feature or edits an illegal region, the part appears to stretch.

We currently support a single application feature (holes) and several skeleton synthesis algorithms (Delaunay Triangulation [19], and *t*-spanners [17, 18]). A connectivity graph is computed without regard for illegal regions, then a path is planned around them for each edge in the graph. This planning is done by computing CSPACE (illegal regions are grown by half the spar width plus the clearance), and its visibility graph [14]. Dijkstra's algorithm [22] is then used to find the shortest path for each connectivity edge. The software is implemented entirely in C++, with the exception of the Forms [23] and Numerical Recipes [24] C source code libraries.

5 Research Issues

Numerous research topics have been discussed in the preceding text. We summarize them here, along with a few not previously mentioned. First, there are a number of data structures, algorithms and heuristics from computational geometry that will be important for skeleton synthesis. These include Delaunay triangulation, spanner graphs, steiner trees, and visibility graphs. To support a system that can provide rapid feedback to the user will require changing existing algorithms. By exploiting the temporal and local coherence inherent in the interactive process, we hope to be able to devise new data structures that support fast incremental updates. Other adaptations include extending current algorithms to work in the presence of illegal regions, and modifying visibility graph techniques to support the synthesis of tapered paths.

Simultaneous design of multiple parts is a difficult issue. We have proposed two approaches (prioritizing and partitioning), but there are many complex issues to be understood.

In addition to skeleton and material synthesis, there are a number of new research topics that dovetail with our general paradigm. Part shape optimization based on finite element analysis is related and could use a synthesized shape as a starting point. On-line product databases with interface geometry have been suggested in the literature.

It seems that shape synthesis could be a powerful aid to designing assembly plans in parallel with the parts to be assembled. The assembly planner and the shape synthesis engine should negotiate and adjust their designs so that the synthesized parts always admit a rational assembly sequence.

6 Conclusion

We have presented a blueprint for what we believe will be a genuinely useful new kind of CAD system. Such a system will have clear economic advantages: Shape synthesis can be compared with automatic placement and routing of library cells that has proven indispensable in the world of CAD for integrated circuit design. Minimizing the stream of input required from the designer will not only speed up the design cycle, it will also facilitate vastly more thorough exploration of the design space than is possible with current CAD systems.

In this design paradigm, conceptual design is more naturally supported because arbitrary geometric details are not hard coded into the design. A measure of functional intent is encoded by modelling only application features and augmenting them with relationship information. This functional level requires less information to unambiguously specify than does explicit geometric construction, and is less subject to change over the life of the design. The representation of the design can be thought of as "live," i.e. that the design represents an infinite family of designs that satisfy the design criteria. Further, since it is the computer that designs the specific part geometry, manufacturing process knowledge can be applied at the part synthesis stage to yield parts that are cheap and easy to manufacture. Global part properties, such as manufacturing process or style of construction, may be changed with relative ease at any time during the design cycle.

An important feature of our view of shape synthesis is that it is, to some extent, separable from other design issues. Shape synthesis can be provided by an engine that may be equally well used as a backend to a direct user interface or as an agent in a system of collaborating software agents. Second, shape synthesis can be decoupled from the choice of data structure used to specify application features. We need only that required and illegal regions be extractable. Finally, shape synthesis is separable from issues of motion of the parts of the machine if we assume that complete descriptions of the nature of part motions are available. Thus shape synthesis can proceed in a static environment.

7 Acknowledgements

Research supported in part by the U.S. Army Research Office under grant number DAAL 91-G-0191, and by IBM under a Manufacturing Research Fellowship.

References

[1] T. Smithers, "AI-based design versus geometry based design, or why design cannot be supported by geometry alone," *Computer Aided Design*, vol. 21, pp. 141–150, April 1989.

[2] J. J. Cunningham and J. R. Dixon, "Designing with features: The origin of features," in *Intl. Computers In Engineering Conference*, pp. 237-243, Aug. 1988.

[3] J. J. Shah, "Assessment of features technology," *Computer Aided Design*, vol. 23, pp. 331–343, June 1991.

[4] J. J. Shah, S. Sen, and S. Ghosh, "An intelligent cad environment for routine mechanical design," in *Intl. Computers In Engineering Conference*, pp. 111–117, American Society of Mechanical Engineers, 1991.

[5] M. R. Cutkosky, D. R. Brown, and J. M. Tenenbaum, "Working with partially specified designs in concurrent product and process design," Tech. Rep. 19891214, Center for Design Research, Stanford University, Stanford, CA, 1989.

[6] V. Shapiro and H. Voelker, "On the role of geometry in mechanical design," *Research in Engineering Design*, vol. 1, no. 1, pp. 69–73, 1989.

[7] M. R. Duffey and J. R. Dixon, "Automating extrusion design: A case study in geometric and topological reasoning for mechanical design," *Computer Aided Design*, vol. 20, pp. 589–596, Dec. 1988.

[8] P. V. Graham and K. T. Ulrich, "Structural synthesis of sheet metal parts: An analogy to path planning using manufacturability as a guide," in *Advances in Design Automation*, pp. 289–294, Vol. 1, American Society of Mechanical Engineers, Sept. 1989.

[9] K. Shimada and D. Gossard, "Automated shape generation of components in mechanical assemblies," in *Advances in Design Automation*, pp. 51–58, Vol. 1, American Society of Mechanical Engineers, 1992.

[10] F. Arbab, *Requirements and Architecture of CAM oriented CAD systems for Design and Manufacture of Mechanical Parts.* PhD thesis, University of California, Los Angeles, 1982.

[11] J. D. Williams, "STICKS—a new approach to LSI design," Master's thesis, Massachusetts Institute of Technology, 1977.

[12] H. L. Johannesson, "Computer aided part design based on standard component interface geometry," in *Advances in Design Automation*, pp. 347–352, American Society of Mechanical Engineers, 1991.

[13] J. J. Craig, *Introduction to Robotics.* Reading, MA: Addison-Wesley, second ed., 1989.

[14] J.-C. Latombe, *Robot Motion Planning.* Norwell, MA: Kluwer Academic Publishers, 1991.

[15] Z.-K. Ling and T. R. Chase, "A technique for the design of an interference free complex planar mechanism," in *Advances in Design Automation*, pp. 433–441, American Society of Mechanical Engineers, 1991.

[16] F. Hwang and D. Richards, "Steiner tree problems," *Networks*, vol. 22, pp. 55–89, Jan. 1992.

[17] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, "On sparse spanners of weighted graphs," *Discrete & Computational Geometry*, vol. 9, no. 1, pp. 81–100, 1993.

[18] J. S. Salowe, "Constructing multidimensional spanner graphs," *International Journal of Computational Geometry & Applications*, vol. 1, pp. 99–107, June 1991.

[19] F. Preparata and M. Shamos, *Computational Geometry.* Springer Verlag, second ed., 1989.

[20] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[21] Parametric Technology Corporation, 128 Technology Drive, Waltham, MA 02154, *Pro/ENGINEER Modeling User's Guide*, 1993.

[22] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Cambridge, Mass.: MIT Press, 1989.

[23] M. Overmars, "The forms library: a package for building graphical interfaces on silicon graphics computers." Public Domain Software Library, ftp://sol.cs.ruu.nl/pub/SGI/FORMS/forms2.3.tar.Z, 1995.

[24] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing.* New York, NY: Cambridge University Press, 1st ed., 1988.