Feature-based Similarity Assessment of Solid Models*

Alexei Elinson
Department of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742

Dana S. Nau
Department of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742

William C. Regli†
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

This paper presents our initial efforts to develop a systematic approach for assessing the similarity of solid models based on how they will be manufactured. The goal of this work is to develop methods that, given a solid model representing the design of a new product, query a product information database (of solid models, associated manufacturing plans, and related attributes) and identify existing designs with manufacturing plans similar to some reasonable plan for the new design—or useful as a starting point for creation of a new plan for the new design.

Our approach is based on the automatic generation (from CAD models) of graph structures that contain manufacturing information (in the form of manufacturing features). We are developing ways to measure similarity among these graph structures, so that given the graph structures corresponding to two different designs, we can tell how similar or different they are. The similarity measure will be used as a basis for indexing and retrieving similar designs from databases. An implementation of our approach is discussed.

We believe our work is a first step in producing computer-generatable and computer-interpretable similarity assessment techniques that will be useful for applications such as variant and hybrid variant/generative process planning systems, indexing schemes for large part inventories, access methods for "smart catalogs," and for performing component searches through product catalogs and on the Internet.

*This work is supported in part by NSF grants NSF EEC 94-02384, IRI-9306580, and DDM-9201779, by ARPA grant DABT63-95-C-0037 and ONR grant DABT63-95-C-0037, and by in-kind contributions from Spatial Technologies and Bentley Systems. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funders.

†Current position: Visiting Research Engineer, Engineering Design Research Center, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3980

1 Introduction

Computer-Aided Design and solid modeling have become essential elements in modern design and manufacturing. Computer-interpretable models of product information are ubiquitous in all phases of the product life-cycle. The ability to efficiently save, index, and retrieve solid models and CAD data has become critical in a wide range of applications, including variant process planning systems, indexing schemes for large part inventories, access methods for "smart catalogs," and for performing component searches through product catalogs and on the Internet. In particular, we are interested in the following closely related problems:

- given a collection of designs, which ones are similar to each other—and how similar are they?
- given a design—or a set of characteristics for what kinds of designs we are interested in—how to find and retrieve similar designs from a database?

Current approaches to these problems involve either complicated geometric comparisons that do not measure similarity according to criteria relevant for important application domains such as manufacturing, or the use of simple indexing and classification schemes such as GT codes that do not incorporate enough product information to allow detailed comparisons of similarity among complex designs.

As a solution to these problems, we are developing an approach based on the use of abstract graph-based representations of designs. In particular, we are developing ways to create *design signatures*—graph structures that represent significant design attributes and significant relationships among

Figure 1: Desirable properties of a similarity measure in different application areas.

Design property	Area of application
function	conceptual design, performance analysis
shape	conceptual and detailed design, design for assembly (DFA)
tolerances	detailed design, DFA, design for manufacture (DFM)
material	detailed design, performance analysis, DFA, DFM
manufacturing methods	process planning, DFM, performance analysis

these design attributes. We are also developing ways to measure similarity among these graph structures, so that given the graph structures corresponding to two different designs, we can tell how similar or different they are. The similarity measure will be used as a basis for indexing and retrieving similar designs from databases.

The specific relationships and attributes that are used—as well as the specific criteria for similarity among designs—will depend on the particular application domain at hand. In the current paper, we give examples involving design attributes, relationships, and similar measurements that are relevant in the application domain of process planning for machined parts.

Section 2 describes what a similarity measure is, and discusses previous work. Section 3 outlines our approach, and Section 4 gives examples from the domain of machined parts. Section 5 discusses the experimental validation of the approach, and Section 6 contains concluding remarks.

2 Measuring Similarity

2.1 Intention

Some of the desirable properties for a similarity measure are:

- It should reflect those notions of similarity that will be useful in the application domain in which it is to be used. For example, Figure 1 shows some of the properties of a design that may be important in different application areas.
- It should provide a way to index designs for quick retrieval from a database. For example, if we are developing a new design, we may want to retrieve (from a database) the closest possible existing design, or a number of "relatively close" existing designs.
- It should be "open." In other words, it should be possible to tune it for a particular application by changing some of the details of what it means for two designs to be similar, and still be able to use the same approach and same algorithms.

2.1.1 Group Technology

Group Technology (GT) is probably the oldest and the most popular type of approach to classifying designs. The basic idea—to make "parts of similar shape on specially grouped machines" [2]—and the consequent need for some way of classifying designs according to their manufacturing requirements—is an old technique, and some references to it can be found as early as 1925. However, only after the works of Mitrofanov [9] and Opitz [10] did it become very popular.

In the last 35 years, many classification schemes have been developed, with emphasis in the different areas of design and manufacturing. In each case the basic idea is to capture critical design and manufacturing attributes of a part in an alphanumeric string, or GT code, that is assigned to that part.

The typical GT code [6] consists of two types of positions. In one case, a position describes some global property of the design such as material, size, type, functionality, etc., and its meaning is completely independent of what values are stored elsewhere. In the other case, a position represents some details that are relevant only for certain types of designs, and thus its meaning depends on the values of other positions.

GT classification schemes are essentially tables and rules that help a designer determine the GT code of a part from a drawing manually. One can use a database of the GT codes for design retrieval, variant process planning, and other manufacturing applications. Since the 1980's several researchers (e.g.,[14, 15]) have worked on automating this manual process for classes of machined parts.

GT qualifies as an open method—one checks previous coding work, selects appropriate criteria and construct new codes relevant to the particular domain of designs. Using question trees to define the code provides an efficient way to build an indexing scheme. One issue is the relevance of GT methods to specific real world design retrieval problems. As it has been used during the last 35 years it works—but it has an inherited drawback: describing designs as short strings creates a coarse classification scheme. Moreover, from the beginning GT coding was intended to be human interpretable, hence the typical questions describe somewhat subjective human impressions of 2D drawings. This has caused difficulty in automating the generation of GT codes.

2.1.2 Geometric Approaches to Measuring Similarity

Another possible basis for classifying designs is to use geometric properties of solid and CAD models. Most of today's CAD/CAM systems use either constructive or boundary models to represent solids.

The use of CSG trees as a way to classify designs has two appealing characteristics: the analogy between volumetric CSG primitives and the volumes of material removed by machining operations, and the ready availability of CSG trees as a basic representational scheme in several geometric modelers. However, the approach suffers from two drawbacks. First, the CSG representation for a design is not unique and a robust method for computing a unique CSG representation for a design has not yet been found (many believe that such a method simply cannot be found [8]). Second, the CSG primitives that would be involved in such a representation do not necessarily correspond to the manufacturing operations that would be used to manufacture the design—and thus the classification might not be very useful for manufacturing practice. As far as we know, no methods to measure similarity on the basis of CSG trees were developed.

Wysk et al [16] have described a similarity measure for solids based on properties of their boundary representations. The approach involves representing a polyhedral approximation of a solid using a graph, in which the nodes correspond to faces of a solid and have labels capturing the faces' orientation and area, and the edges correspond to the adjacency relation between solid faces, and are labeled by the corresponding solid angles. To compare two solids they use a sophisticated algorithm to take the graphs of these solids and map them into each other in such way that the area and orientation of corresponding nodes are as close as possible. The results of such mapping are expressed as a real number in a range from 0 to 1. As a new measure of "relaxed" geometrical similarity their work looks very interesting, but there are several difficulties to be overcome before it can be useful as a classification scheme for manufacturing:

- In its current form, the method works only with polyhedral objects—any non-planar faces of the designs must first be replaced with planar approximations. This may cause difficulty in classifying solids with a significant number of cylindrical or sculptured surfaces.
- The measure of similarity is not symmetrical (similarity between solids A and B is not equal to the similarity between B and A). This will cause difficulties in using it as the basis for a traditional database indexing scheme, since such schemes assume a symmetrical measure.
- In its current form, the method does not incorporate (or reflect) manufacturing considerations, such as approachability, fixturing, and operation interference, and we do not see any obvious way to add them.

2.1.3 Feature-Based Similarity

During last several years, a number of efforts have been made to develop algorithms to examine CAD designs and extract features that correspond to manufacturing operations. This creates an opportunity for a third kind of approach to measuring the similarity of CAD designs: to automatically generate representations of them based on their manufacturing features, and compare the similarity of these feature-based representations. Such an approach is the subject of this paper.

3 Approach

3.1 Equivalence Hierarchy

Suppose we are given a class of designs $\mathcal{D}$, and want to measure the similarity of various designs in $\mathcal{D}$. We will judge the similarity of two designs d and d' by examining various properties of these designs—things that are either specified explicitly in the design, or can be deduced by examining it. One example (although certainly not the only one!) would be the kinds of properties that one might examine in order to compute a GT coding scheme.

If the set of properties we are examining is $P = \{p_1, p_2, \ldots, p_n\}$, then we will use $p_i(d)$ to denote the value of the property p_i for the design d. For each property p_i, we can define an equivalence relation E_i such that $E_i(d, d')$ is true if and only if $p_i(d) = p_i(d')$. For example, if $p_i(d)$ is the material from which d is to be made, then $E_i(d, d')$ will be true only if d and d' are to be made from the same material.

Given a design d, suppose we want to find designs in $\mathcal{D}$ that are "relatively similar" to d, or find the design that is "most similar" to d. For such a task, we need a way to measure the similarity of two designs along some numeric scale—so that we can say, for example, that d is *more* similar to d' than it is to d''. Below, we define a way to construct such a measure from the equivalence relations $\{E_i\}$ defined above.

Let R_0 be the equivalence relation whose only equivalence class is $\mathcal{D}$ itself; and for $i = 1, \ldots, n$, let

$$R_i = E_1 \wedge \ldots \wedge E_i;$$

i.e., $R_i(d, d')$ is true only if $E_1(d, d')$, $E_2(d, d')$, $\ldots$, $E_i(d, d')$ are all true. Then for each i, every equivalence class of R_{i+1} is a subset of some equivalence class of R_i; i.e., if $R_{i+1}(d, d')$ is true, then $R_i(d, d')$ will also be true. Thus $H = \{R_0, \ldots, R_n\}$ is a sequence of equivalence relations that make progressively more detailed distinctions among designs. We call H an *equivalence hierarchy*.

If d and d' be two designs in $\mathcal{D}$, then we define their *measure of similarity* to be

$$M(d, d') = \text{the largest } i \text{ such that } R_i(d, d') \text{ is true.}$$

If $M(d, d') = i$ for some i, then we say that d and d' are *i-similar*. If d is a design, $D \subseteq \mathcal{D}$ is a set of designs, and d^* is a design in D, then we say that d^* is *closest* to d among the designs in D if for every design $d' \in D$,

$$M(d^*, d) \geq M(d', d).$$

3.2 Classification Trees

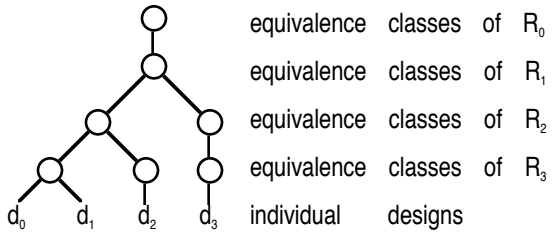


Figure 2: An example of a classification tree. Note that the measure of similarity of any pair of designs is the depth of their deepest common ancestor—for example, $M(d_0, d_1) = 3$, $M(d_0, d_2) = 2$, and $M(d_0, d_3) = 1$.

Let $H = \{R_0, \ldots, R_k\}$ be an equivalence hierarchy over some class of designs $\mathcal{D}$, and let D be a collection of designs such that $D \subseteq \mathcal{D}$. Then the *classification tree* for D under H is the tree T whose nodes are all of the designs in D and all of the equivalence classes of all of the relations in H; and whose edges are the following:

- for each $i > 0$ and each equivalence class E of R_i, an edge between E and the equivalence class of R_{i-1} that contains it;
- for each design in D, an edge between D and the equivalence class of R_k that contains it.

As an example, suppose that D consists of four designs $\{d_0, d_1, d_2, d_3\}$, and H consists of four equivalence relations R_0, R_1, R_2, R_3 whose equivalence classes are as follows: R_0 and R_1 each have a single equivalence class which is D itself; R_2's equivalence classes are $\{d_0, d_1, d_2\}$ and $\{d_3\}$; and R_3's equivalence classes are $\{d_0, d_1\}$, $\{d_2\}$, and $\{d_3\}$. Then the classification tree for D under H is as shown in Figure 2.

Note that T has the following properties:

1. for each i, the equivalence classes of R_i are all at depth i of the tree;
2. the leaves of T are all at depth $k + 1$ and are all of the designs in D;
3. for each equivalence class E, the designs of D that are in E are the leaves of the subtree below E;
4. for any two designs $d, d' \in D$, $M(d, d')$ is the depth of their deepest common ancestor in T.

Such a tree can support the quick performance of queries such as the following:

- If we want to find all designs in D that are i-similar to some design d in D, they will be consist of all leaves below the ancestor of d at level i of the tree. The amount of time required to do this will be $O((k - i)n)$, where n is the number of designs that are i-similar to d.
- If we want to find all designs in D that are closest to some design d in D, they will be represented by all leaves below the deepest ancestor of d that has more than one child. The amount of time required to do this will be $O(kn)$, where n is the number of designs that are closest to d.

If we want to find all designs in D that are i-similar or closest to some design d that is not in D, we can do so by inserting d into the tree T, and then using one of the methods described above.

3.3 Design Signatures

Suppose we are given a set D of designs stored in a database, and want to create an equivalence hierarchy and a classification tree as described above. To do this, we will need to decide what design properties we consider relevant, and devise appropriate ways to store and retrieve the property values for each design in the database.

Some of the properties that we may wish to represent—e.g., the material from which the design will be made, or whether it is a rotational or prismatic part—may be "global" properties of the design that are similar to the kinds of design properties used in GT coding schemes, and can be represented by attaching a simple list of tags to the design in the database. Other properties that we might wish to represent—for example, the tolerance constraints associated with various manufacturing operations—may be difficult to represent without referring to specific portions of the design's geometry. In some cases it may be convenient to represent this information as a set of annotations to the CAD model—but in other cases a more abstract representation of the design may be needed. For such purposes, we propose to use what we will call a *design signature*: a graph structure containing nodes that represent various attributes of the design, and edges that are labeled to represent various relationships among those attributes.

The specific attributes and relationships among them will depend the particular application domain at hand. As an example, suppose we are interested in process planning for machined metal parts. In this case, to represent information about the machining operations to be used to create a design, some of the nodes in the design signature might correspond to the tool-swept volumes for various machining operations that might be used in machining the part; and some of the relationships among those nodes might correspond to notions of inteference among those operations. Although it might not be immediately obvious to the reader how to derive such information directly from the CAD model, Section 4.1 discusses ways to do this.

In many application domains, testing for similarity among two different designs may often require performing subgraph-isomorphism tests on their signature graphs. As another example from the machining domain, if we are given a design d and want to find a design d' that has a similar process plan, we may want to look for a one-to-one correspondence between

the features in d's feature-based model and the features in d''s feature-based model.[1] Since the graph isomorphism problem can be quite difficult computationally, it may be quite useful to keep the feature-based model as simple as possible, by omitting unnecessary detail. However, if we do this, then the feature-based model may lack some of the detail needed for more detailed comparisons later on.

The above considerations suggest that for each design d, we may wish to have several different signatures, each representing d at a different level of abstraction, so that we can use the more detailed signatures to represent the information needed for detailed comparisons of d with other designs, and the more abstract signatures for less detailed comparisons. Although creating these design signatures may be time-consuming, we will only need do it once for each design d; and once we have done it, we will be able to use the results again and again, each time we need to perform a comparison between d and some other design d'.

More specifically, suppose our equivalence hierarchy is $H = \{R_0, \ldots, R_k\}$, with $R_i = R_{i-1} \wedge E_i$ for each $i > 0$. Then to represent each design d we might use a sequence of design signatures $S_1(d), \ldots, S_k(d)$, where $S_i(d)$ is the signature that we will use to compute the equivalence relation E_i. In general, E_{i+1} will need to make more detailed comparisons between d and d' than E_i does. If we can make these comparisons using the same signature graph then $S_{i+1}(d)$ and $S_i(d)$ will be identical; otherwise $S_{i+1}(d)$ will be more detailed than $S_i(d)$.

The most detailed signature for d is the signature $S_k(d)$, which we will call d's *basic signature*. To generate the signatures $S_{k-1}, S_{k-1}, \ldots, S_0$, we will usually start with the basic signature and use various pruning heuristics that remove nodes and edges to produce simpler and simpler signature graphs. The precise nature of these heuristics depends on the specific manufacturing domain.

4 An Example

In the previous section, we described an approach for representing design information and using it to classify designs. Because the approach is quite general, our description was rather abstract. To make the idea more concrete and show how it might be useful in practical domains, we now describe a specific instantiation of this approach, in a prototype system for classifying machined parts for use in process planning. For a more detailed description, see [1].

4.1 Manufacturing Features

There have been several attempts to arrive at precise definitions for manufacturing features. Most have involved the use of *form features*: local geometric configurations on a manufactured part that have some engineering significance during the part's lifetime.

The approach we adopt is to use a particular kind of form feature that we call *a machining feature*, that corresponds directly to the volume of space swept by the cutting tool in a machining operation. To perform a machining operation, one sweeps the cutting tool along some trajectory. This swept volume can be divided into two parts: the *removal volume*, which is the volume of material that is to be removed by the machining operation; and the *accessibility volume*, which is the non-cutting portion of the tool-swept volume [4, 12]. The machining features described below represent the removal volume and the approach direction.

A *feature type* is a parameterized volumetric template M that represents the removal volume of a particular type of machining operation. An *instance*, f, of a machining feature is created by a particular machining operation with a single cutting tool in one tool setup.[2] M represents the idealized feature template and f is a particular example of an operation that can occur with respect to some part. We will sometimes use the term *machining feature* to refer to a feature type, and sometimes to refer to a feature instance; the meaning should be clear from context. A **primary machining feature** is one that contains as much of the stock as possible without intersecting with the part, and as little space as possible outside the stock; our reasons for using primary features are discussed in [4].

In general, there may be several alternative interpretations of the part as different collections of machining features, each interpretation corresponding to a different way of manufacturing it. A *feature-based model* is a set of feature instances that models a single, unique interpretation of the part. The feature recognition problem is defined as follows: given a collection of machining features, $\mathcal{M} = \{M_1, M_2, \ldots M_j\}$, a part P, and a piece of stock S, find the set $\mathcal{F}$ of instances of feature types from $\mathcal{M}$ recognized from P and S. The *feature set* $\mathcal{F}$ is a finite set of features; the set being composed of the union of the alternative feature-based models for the part.

Techniques exist for automatically extracting machining features from CAD models. For this purpose, we are employing the trace-based approach developed by Regli et al. [11, 12] to obtain drilling and milling features from solid models. In this approach, the initial workpiece, S, is represented as a solid model of raw stock material to be acted upon by a set of machining operations. The machined part is a solid object, represented by a solid model of the part P, to be produced as a result of a finite set of machining operations. The *delta volume* is the regularized difference [5] of the initial workpiece and the part: $\Delta = S -^* P$. A *trace* corresponds to the information contributed to P and Δ by an instance of a feature f of type M and provides sufficient

[1] Although our graphs have labels on the edges and nodes, graph isomorphism is nearly the same as for ordinary unlabeled graphs. The only difference is that we require corresponding edges and nodes to have the same labels.

[2] In practice, f, when considered with respect to a particular set of manufacturing resources, may map to several individual machining operations (i.e., such as drill-ream-bore for a particular drilling feature).

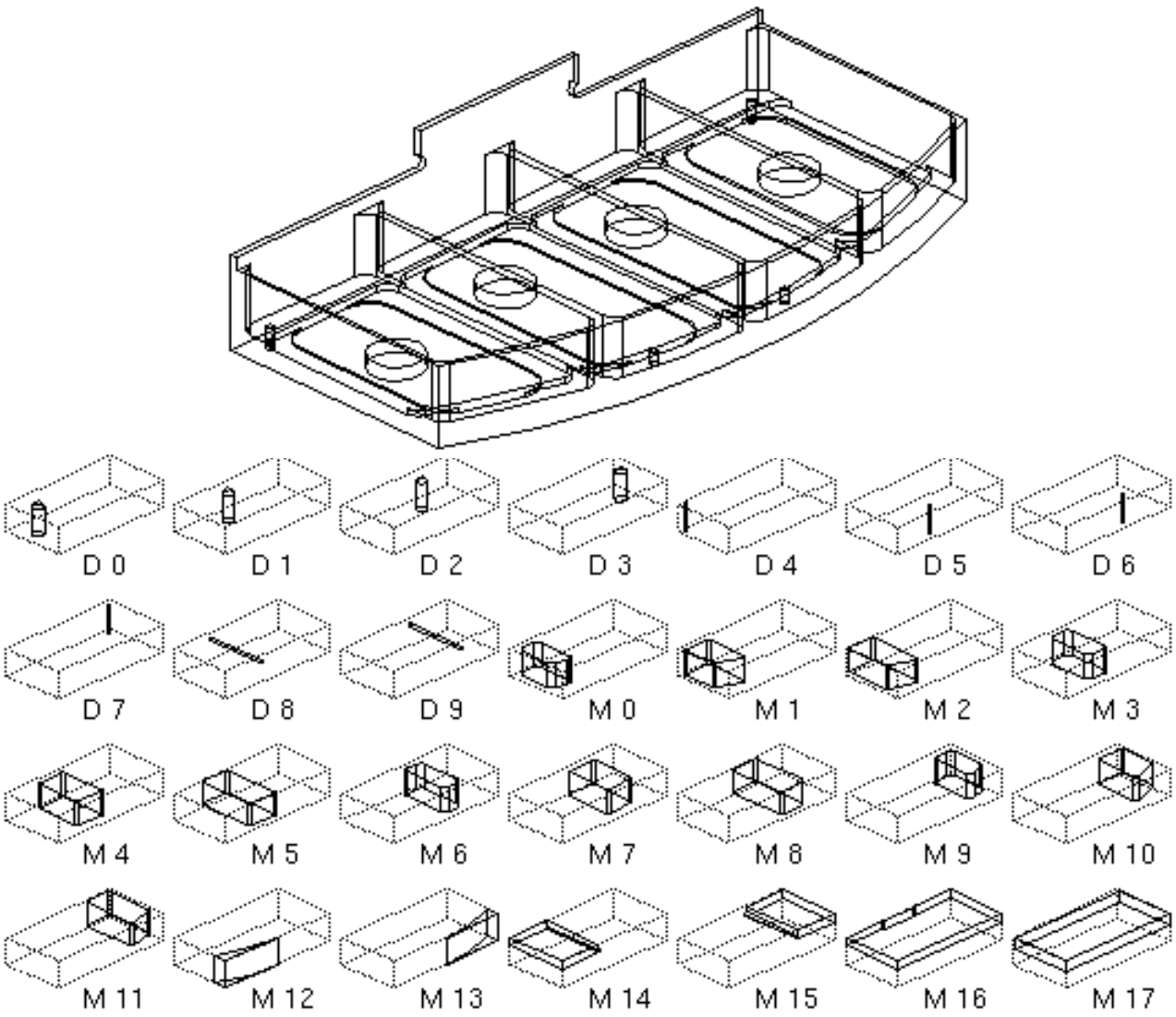


Figure 3: A prismatic part that can be manufactured using milling and drilling operations, and the machining features extracted from its CAD model. $D0$ through $D9$ are drilling features, and $M0$ through $M17$ are milling features. There are also ten other drilling features $D0'$ through $D9'$ that are not shown here; these features have the same removal volumes as $D0$ through $D9$ but the opposite approach directions.

information for calculating the parameters of f. Given P and S, the output of the recognition system is a finite set of machining features making up the *feature set* for P and S. An example appears in Figure 3.

4.2 Basic Signature

To represent machining features in the basic signature, we use a graph whose nodes correspond to features and whose edges correspond to relationships among the features, with labels on the nodes and the edges giving various parameters that may be useful for classification purposes. It is tempting to try to represent as many feature parameters as possible—but without further experimentation it is impossible to say which feature parameters will be the most useful. In our prototype system we have chosen to represent only those feature parameters that are obviously important or are easy to calculate.

4.2.1 Node Parameters

Here are some of the parameters for all features that we represent in the basic signature. All of this information is calculated directly from the CAD model and the feature information:

- what type of feature it is (i.e., milling feature or drilling feature);
- the possible approach directions for machining the feature, and the depth of the material removal volume from each direction;
- the faces of the stock that the feature touches;
- the feature's maximal possible cutting-tool radius;
- whether or not the feature is mandatory (i.e., whether or not it removes some volume of material that is removed by no other features).

For drilling features, we also represent the tip angle of the cutting tool (calculated by examining the conical face that the tool creates in the design) and relationships between features and design. For milling features, we represent various information about the profile swept by the milling tool.

4.2.2 Edge Labels

In general, we use edges between features to represent interactions among the features that may affect how they will be machined. Currently, the only kind of interaction we represent between two features is whether they intersect.[3] If two features intersect, we represent various properties of the intersection, such as the ones described below.

[3] In future work, we intend to include interactions that result from tolerancing constraints such as datum dependencies, and interactions that result from how the workpiece will be fixtured.

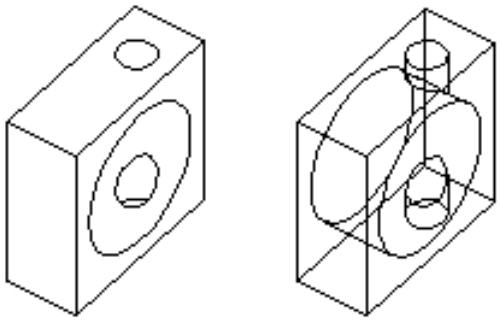
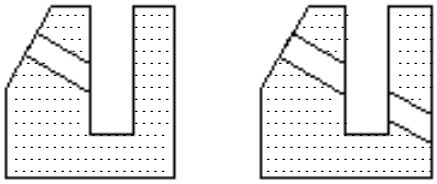


Figure 4: An example of a critical intersection between two drilling features.



(a) non-critical intersection (b) critical intersection

Figure 5: Examples of intersections between a drilling feature and a milling feature.

We will say that a feature intersection is *critical* if it can create manufacturing precedence constraints among the machining operations used to create the intersecting features or if it will require some special processing (such as facing operations) that would not otherwise be needed; otherwise we will say that it is *non-critical*. To decide whether an intersection is critical, we use several criteria based upon manufacturing heuristics from [3, 18]. Here are simplified versions of some of our criteria:

- Any volumetric intersection between two milling features is non-critical. Rationale: the milling features found by our feature extraction module are guaranteed to be accessible for machining, and thus there are no approachability constraints for milling features.
- An intersection of two orthogonal drilling features is critical if one of the drilling operations can create a workpiece such as the one shown in Figure 4, in which the second drilling operation requires the drill bit to enter the material through a surface that is non-orthogonal or incomplete. Otherwise it is non-critical.
- If there is an intersection between a milling feature and a drilling feature in which the milling feature can create a workpiece in which the drill bit would enter a surface that is non-orthogonal or incomplete, then the intersection is critical; otherwise it is non-critical. See Figure 5 for examples.

In addition to critical and non-critical intersections, there are several other properties that we represent, such as the following:

- If two features have exactly the same removal volume, then we call them *siblings*. We represent this because one would not normally want to machine both of these features.
- If two features have the same approach direction, then we say that they are *setup-compatible*. We represent this because (provided that various other restrictions are satisfied), one can machine them during the same setup, which may be important for minimizing the number of setups in the process plan.
- If two drilling features have the same radius and their length-to-diameter ratios fall within certain intervals, then we say that they are *tool-compatible*. We represent this because it may be possible to make them using the same drilling tool, which may be important for minimizing the number of tool changes in the process plan.
- If two features f_i and f_j have the same approach directions and neither is a subset of the other, but f_i's approach face is contained within f_j, then we will say that that f_i is a *child* of f_j. Intuitively, this corresponds to the precedence ordering that one would usually use in practice.

4.2.3 Example

Figure 6 shows a simplified version[4] of the basic signature for the design shown in Figure 3. Each edge is labeled with an "N" if the intersection is non-critical, and with a "C" if it is critical. Whenever one feature is a child of another, there is a directed edge from the parent to the child, labeled with a "P". Whenever one feature contains another, there is a directed edge from the larger feature to the smaller one, labeled with an "S". Whenever two features are identical except for their approach directions, the edge is drawn in boldface.

4.3 Building a Signature Sequence

To represent design d at various levels of abstraction, we would like to create a sequence of signatures $\{S_k(d), \ldots, S_1(d)\}$ for use in computing the equivalence relations $R_k, \ldots, R_1$, respectively; where S_k is the most detailed *basic signature* and for each i, S_{i-1} is either exactly equal to S_i or is more abstract.

There are several ways to make a signature graph more abstract. Due to space constraints we cannot describe them in detail here, but here are some simple examples:

- Remove some labels or replace them with more general ones. The rules for doing this depend heavily on the particular domain, and probably cannot be generalized. One example would be to drop labels specifying some of the geometric details of features, such as the profile of a milling feature or the tip angle of a drilling feature.
- Find two nodes with the same basic parameters such that one is a child of the other. Replace them with a single node having the same basic parameters, edges, and edge labels as the original two nodes. For example, in Figure 7, M_1 is a child of M_0, and both are milling features with similar parameters, so we could replace them with a single node that is connected to all of the nodes that M_0 and M_1 were connected to.
- Find subgraphs that are similar, and remove all but one of these subgraphs from the signature graph. For example, in the signature $S_k(d_4)$ of Figure 8, the subgraph whose node set is $\{D_0, D_1, M_0\}$ is similar to the subgraph whose node set is $\{D_2, D_3, M_1\}$, so we could remove one of these subgraphs.
- Find a group of unconnected nodes that have the same basic parameters, and replace them with a single node having the same parameters, edges, and edge labels as the old nodes. For example, in Figure 9, the two milling feature M_0 and M_1 are identical except for their locations, so we could replace them with a single node that is connected to both $D0$ and $D1$.

4.4 Comparing Signatures

To compare two designs d and d' in the worst case—when they are geometrically equal—we have to perform r isomorphism checks, where r is the number of different signatures in the signature sequence. At first glance, this might seem prohibitively expensive, since the best known algorithm for checking graph isomorphism takes exponential time in the worst case. However, in our experiments our approach generally performed quite well, for the reasons described below:

1. The general approach for checking isomorphism is quite simple [7]: compare parameters describing graphs as a whole (number of nodes and edges, average degree of the node, etc.), then partition all nodes of each graph into groups so that any possible mapping between graph vertices will map all nodes of any group into all nodes of the corresponding group, and then apply brute-force methods to all nodes with respect to these groups.

   For "classical" undirected graphs without labels on the edges, it is usually impossible to partition the nodes into small groups—but in constrast, our heavily labeled nodes and edges usually permitted our partition algorithm to achieve very good granularity, so that the following brute-force stage usually did not require very much time in our experiments.

[4] To keep the figures simple, we have omitted all nodes other than those that represent machining features, all node parameters other than the type of machining operation to be used, and all edge labels other than critical and non-critical interferences.

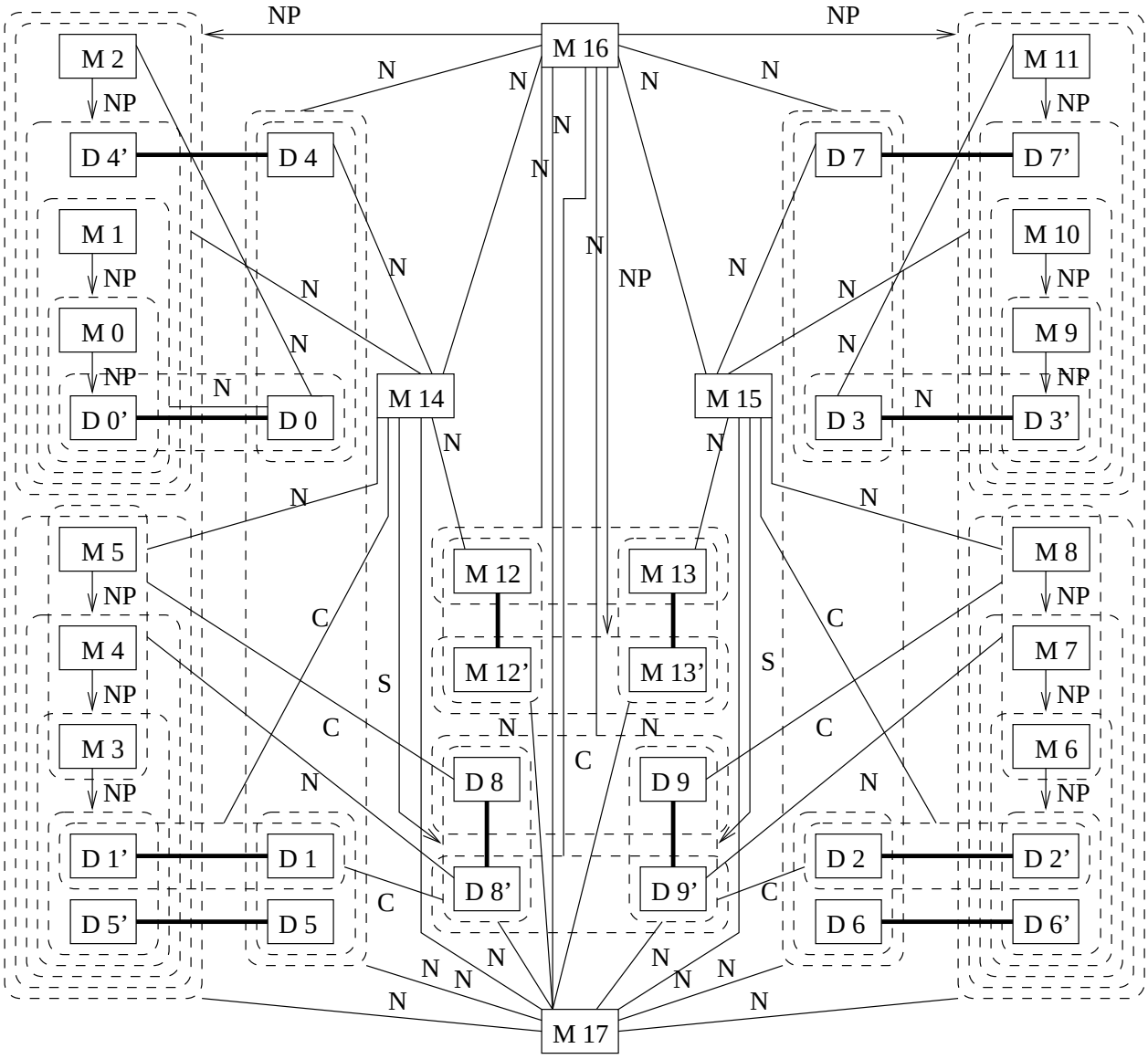
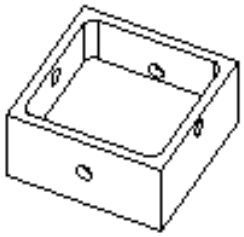
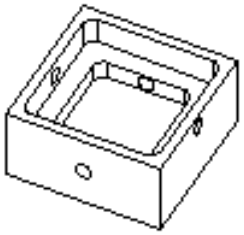


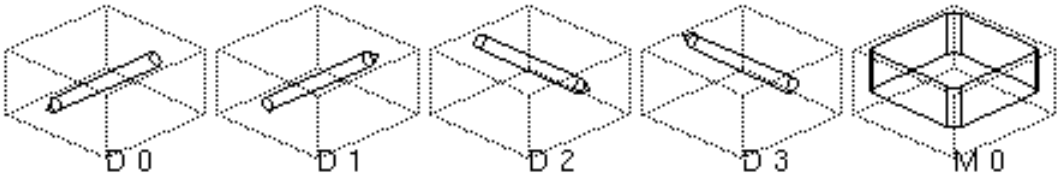
Figure 6: A simplified version (see footnote 4) of the basic signature of the design shown in Figure 3.



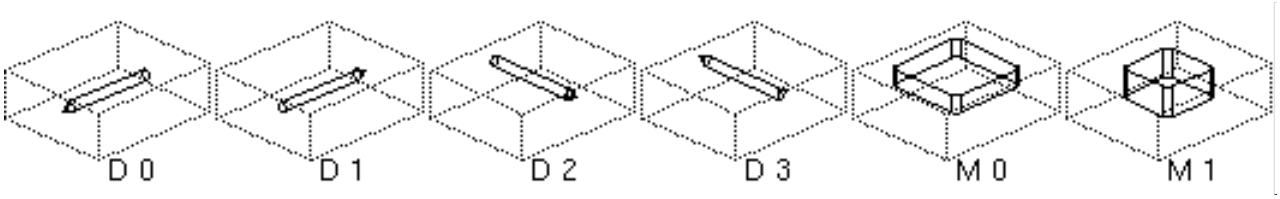
(a) a design d_1



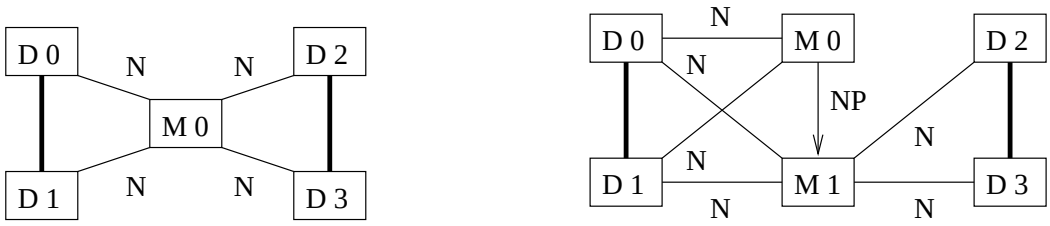
(c) a design d_2



(b) d_1's machining features

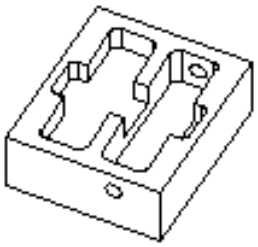


(d) d_2's machining features

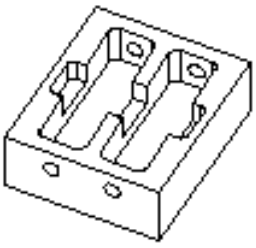


(e) simplified versions (see footnote 4) of the basic signatures $S_k(d_1)$ and $S_k(d_2)$

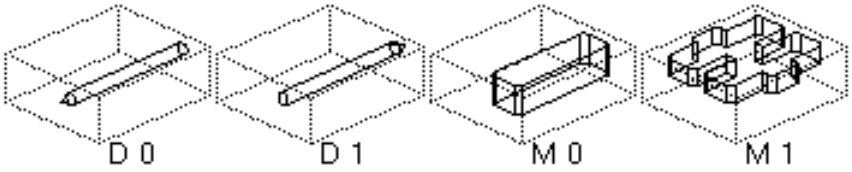
Figure 7: Two similar designs d_1 and d_2. If we replace M_0 and M_1 with a single node that is connected to all of the nodes that M_0 and M_1 were connected to, the resulting signature $S_{k-1}(d_2)$ is isomorphic to $S_k(d_1)$.



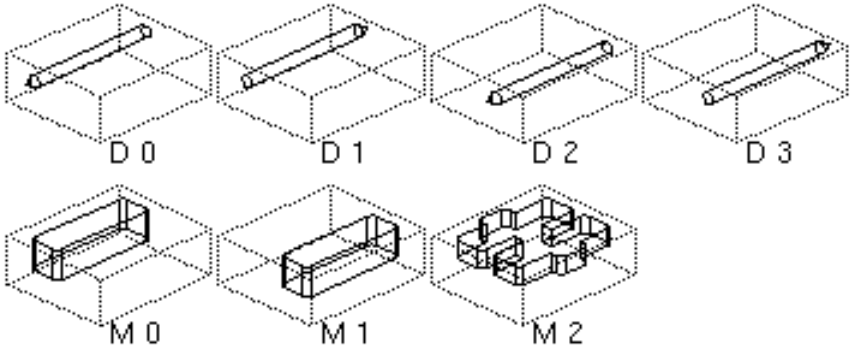
(a) a design d_3



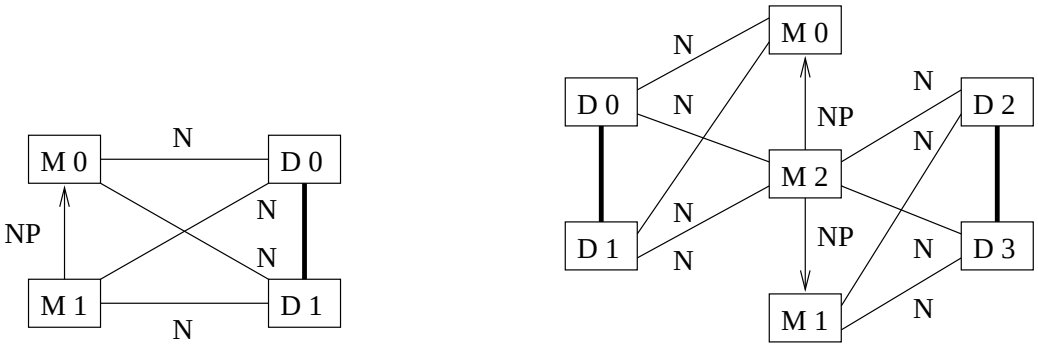
(c) a design d_4



(b) d_3's machining features

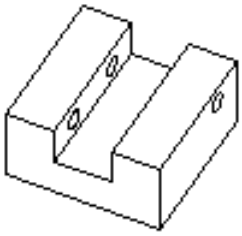


(d) d_4's machining features

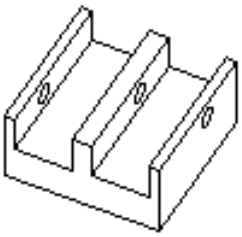


(e) simplified versions (see footnote 4) of the basic signatures $S_k(d_3)$ and $S_k(d_4)$

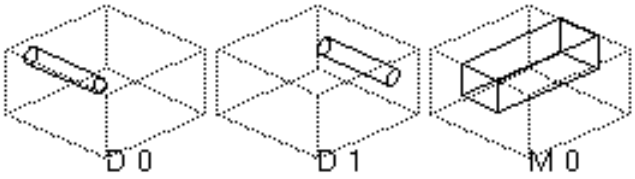
Figure 8: Two similar designs d_3 and d_4. If we remove either the nodes D_0, D_1, M_0 or nodes D_2, D_3, M_1, the resulting signature $S_{k-1}(d_4)$ is isomorphic to $S_k(d_3)$.



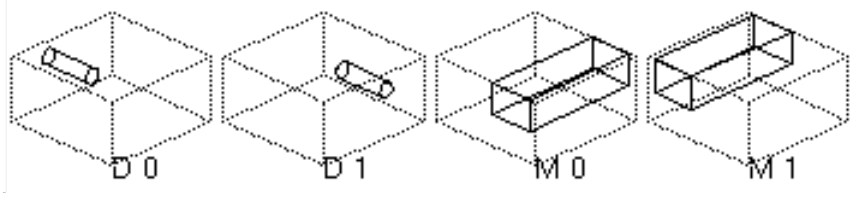
(a) a design d_5



(c) a design d_6



(b) d_5's machining features



(d) d_6's machining features

(e) simplified versions (see footnote 4) of the basic signatures $S_k(d_5)$ and $S_k(d_6)$

Figure 9: Two similar designs d_5 and d_6. If we replace M_0 and M_1 with a single node that is connected to both $D0$ and $D1$, then the resulting signature $S_{k-1}(d_6)$ is isomorphic to $S_k(d_5)$.

The partition algorithm itself is quite expensive (rough estimation gives $O(v^3 \log v)$, where v is a number of nodes in the graph), but the partitions only need to be computed once for each signature graph during its entire lifetime—thereafter, they can be stored along with the signature graph and retrieved as needed. Thus, the time complexity of the partition algorithm is less important than it would be otherwise.

2. Signature graphs permit significantly more sophisticated methods to check that they *can* be isomorphic. For "classical" graphs, usually one can check only such parameters of the graph as the number of nodes with different degrees, but for signature graphs such statistics can be collected about each type of label and all of their combinations if necessary.

3. In the process of signature sequence comparison we begin by comparing the most abstract graphs $S_1(d)$ and $S_1(d')$, and only compare the most complex graphs $S_k(d)$ and $S_k(d')$ at the very end. Even if we end up checking all of $S_1, \ldots, S_k$, in the worst case this takes the same big-O time as if we had checked only S_k. In most cases we will find that $S_j(d) \neq S(j(d'))$ for some $j < k$, and exit without ever checking the basic signature S_k. This saves a great deal of time: in the worst case it takes exponentially less time to check $S_0, \ldots, S_j$ than to check S_k.

   Furthermore, we can often do much better than the worst case, because we can speed up the brute-force stage of checking each signature S_i using the information about nodes mapping obtained during the isomorphism check of the more abstract signature S_{i-1}.

5 Validation

To validate a new classification system, we wanted show that for a reasonable collection of realistic objects, the techniques correctly and efficiently (1) find identical objects and (2) return reasonably intuitive estimations of similarity between different objects.

One major obstacle is the lack of a generally available large and varied data set of CAD models for mechanical designs. A further complication is the lack of an agreed-upon standard for what it means to be *similar* from the *manufacturing point of view*—answers vary depending on the individual to whom the question is posed. Hence, large-scale validation was not possible and we chose to perform controlled experiments to determine how well the prototype system operates on reasonable examples.

To perform our study, we used a collection of solids taken from the NIST Design, Process Planning, and Assembly Repository[5] and employed the experimental F-Rex [11, 12] feature recognizer previously developed at the University of Maryland at College Park.

The prototype system used few feature parameters, but still was able to identify identical designs and its estimations of similarity well corresponded to the design families. The experiments, once pre-processing was performed, ran in moderate user-time and we did not observe that any computational bottlenecks were created by the isomorphism checks. Interestingly as well, we noticed that the approach worked even on those parts where the F-Rex feature recognizer had difficulty or produced spurious results. We believe that later versions of the implementation can be enhanced through using the design features in the CAD model and integration with a commercially tested features tool.

6 Conclusions

This paper has described our approach for automatically assessing the similarity of CAD models. Our approach involves automatically generating graph structures called *design signatures* (abstract representations of the design that contain information that is relevant to some application domain), and ways to examine the design signatures to determine similarity among designs.

The approach is intended to be general in the sense that the same basic ideas could be used in several different application domains. However, the information represented in the design signatures, as well as the criteria for judging the similarity of design signatures, is heavily domain-specific.

As an example of a particular application domain, we have focused on process planning for machined parts. In this problem domain, we have found it useful to use design signatures in which much of the information consists of volumetric features that correspond to machining operations, along with various relationships among those features. There are several reasons why such information is useful in the process planning domain:

- since it corresponds to basic operations in the process plan, it will let us develop measures of design similarity that are useful for retrieving designs that have similar process plans;

- Since a number of well known techniques exist for extracting machining features from CAD models [13, 17] (for example, we use the technique described in [12]), it is possible to generate the design signatures automatically.

Our future work has two basic directions.

- We intend to extend the theoretical basis and algorithmic work, by developing new definitions of equivalence relations upon design signatures and new methods of signature simplification. We intend to implement these definitions and algorithms to provide a useful similarity

[5] Available at `http://www.parts.nist.gov`.

measure for machined parts and a practical classification and retrieval system based upon this measure. The latter will require paying close attention to engineering domain knowledge such as the details of various manufacturing processes and the corresponding geometric features, and developing ways to represent feature interactions that result from tolerancing and fixturing constraints.

- We are developing ways to "slice" design signatures into pieces that are meaningful from the point of view of process planning. This will allow us to use our classification techniques to measure the similarity of these design slices rather than designs as a whole. We intend to use this as the basis for a hybrid variant/generative approach to process planning, that retrieves design slices from a database and and combines them together to form process plans for new designs.

References

[1] A. Elinson, D. Nau, and W. C. Regli. Solid similarity measurements. Technical Report ISR-TR96-63, The University of Maryland, 1996.

[2] C. C. Gallagher and W. A. Knight. *Group Technology Production Methods in Manufacture*. Ellis Horwood Limited, Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England, 1986. ISBN 0-470-20294-7.

[3] Satyandra K. Gupta. *Automated Manufacturability Analysis of Machined Parts*. PhD thesis, The University of Maryland, College Park, MD, 1994.

[4] Satyandra K. Gupta, William C. Regli, and Dana S. Nau. Manufacturing feature instances: Which ones to recognize? In Jaroslaw Rossignac, Joshua Turner, and George Allen, editors, *Third Symposium on Solid Modeling Foundations and CAD/CA M Applications*, pages 141–152, New York, NY, USA, May 17-19 1995. ACM SIGGRAPH and the IEEE Computer Society, ACM Press. Salt Lake City, Utah.

[5] Christoph M. Hoffman. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers Incorporated, CA, 1989.

[6] A. Houtzeel. Miclass, a classification system based on group technology. Technical Report Working Paper MS #75-721, Society of Manufacturing Engineers, 1975.

[7] J. Kobler, U. Schoning, and J. Toran. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser, 675 Massachusetts Avenue, Cambridge, MA, 02139, U.S.A., 1993. ISBN 0-8176-3680-3.

[8] Y. C. Lee and K. S. Fu. Machine understanding of csg: extraction and unification of manufacturing features. *IEEE Computer Graphics & Applications*, **7**(1):20–32, 1987.

[9] S. P. Mitrofanov. *The Scientific Principles of Group Technology*. National Lending Library Translation, 1966.

[10] H. Opitz. *A Classification to Describe Workpieces*. Pergamon Press, Oxford, 1970.

[11] W. C. Regli, S. K. Gupta, and D. S. Nau. Toward multiprocessor feature recognition. *Computer Aided Design*, 1997. To appear.

[12] William C. Regli, Satyandra K. Gupta, and Dana S. Nau. Extracting alternative machining features: An algorithmic approach. *Research in Engineering Design*, 1995. To appear.

[13] J. Shah, Y. Shen, and A. Shirur. Determination of machining volumes from extensible sets of design features. In Jami Shah, Martti Mäntylä, and Dana Nau, editors, *Advances in Feature Based Manufacturing*, pages 129–157. Elsevier/North Holland, 1994.

[14] J. J. Shah and A. Bhatnagar. Group technology classification from feature-based geometric models. *Manufacturing Review*, 2(3):204–213, 1989.

[15] A. Srikantappa and R. Crawford. Automatic part coding based on interfeature relationships. In Jami Shah, Martti Mäntylä, and Dana Nau, editors, *Advances in Feature Based Manufacturing*, pages 215–237. Elsevier/North Holland, 1994.

[16] T.-L. Sun, C.-J. Su, R.J. Mayer, and R.A. Wysk. Shape similarity assessment of mechanical parts based on solid models. In *Design Engineering Technical Conferences*, volume **83**(2), pages 953–962. ASME, 1995.

[17] J. H. Vandenbrande and A. A. G. Requicha. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1269–1285, December 1993.

[18] Jan H. Vandenbrande. *Automatic Recognition of Machinable Features in Solid Models*. PhD thesis, University of Rochester, Rochester, NY, USA, 1990.