Machining Feature-based Comparisons of Mechanical Parts

Vincent Cicirello
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213-3890
cicirello@ri.cmu.edu

William C. Regli
Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104
regli@drexel.edu

Abstract

Solid Models are the critical data elements in modern Computer-Aided Design (CAD) environments, describing the shape and form of manufactured artifacts. Their growing ubiquity has created new problems in how to effectively manage the many models that are now stored in the digital libraries for large design and manufacturing enterprises. Existing techniques from engineering literature and industrial practice, such as group technology, rely on human-supervised encodings and classification; techniques from the multimedia database and computer graphics/vision communities often ignore the manufacturing attributes most significant in the classification of models.

This paper presents our approach to manufacturing similarly assessment of solid models of mechanical parts based on machining features. Our technical approach is three-fold: (1) perform machining feature extraction to map the solid model to a set of STEP AP 224 machining features; (2) construct a model dependency graph from the set of machining features; (3) find the nearest neighbors to the query graph using an iterative improvement search across a database of other models. We also present empirical experiments to validate our approach using our testbed, the National Design Repository (http://www.designrepository.org).

The contribution of this research is the first fully automated technique for machining feature-based comparisons of mechanical artifacts. We believe that this work can lead to radical changes in the way in which design data is managed in modern engineering enterprises.

1. Introduction

We present an approach to comparing the manufacturing similarity of solid models of machined artifacts based on their machining features. Increasingly, manufacturing enterprises must maintain vast digital libraries and databases of Computer-Aided Design (CAD) and Computer-Aided Process Planning (CAPP) knowledge. Such information includes the parametric solid models of parts and assemblies, as well as Numeric Control (NC) machining programs, production plans and cost data.

Our research goal is to develop methods to interrogate large knowledge-bases of solid models of machined artifact in order to enable variational process planning and cost estimation. This work is part of the National Design Repository project (`http://www.designrepository.org`), which is an ongoing effort to collect and archive "open source" CAD data and solid models and develop data management technologies for handling engineering information.

Given that we have a large Repository we wish to search, and a new solid model to use as a query, our technical approach is three-fold:

1. **Perform feature extraction to map the solid model to a set of STEP AP 224 machining features.**

   If operating inside a CAD environment, one could plan to retain the design features as new models are created. However, as has been often noted, design features are not necessarily in one-to-one correspondence with manufacturing features. For legacy data and for solid models that are converted between modeling systems, there may not be any readily available feature information. Our work uses automatic feature recognition, based on the FBMach[1] System from Allied Sig-

[1]FBMach is the "Feature-Based Machining Husk," and is a copyrighted

nal [4, 20] to generate feature data to be used in the indexing algorithms.

2. **Construct a *model dependency graph* from the set of machining features.**

   Given the set of machining features for an artifact, our model dependency represent an intermediate data structure to be used to model feature interactions and dependencies. Our belief, which this paper empirically tests, is that machined parts with similar feature sets and similar feature interactions have a high probability of possessing similar manufacturing plans. Model dependency graphs capture the feature interactions in a given set of features.

3. **Find the nearest neighbors to the query model among those in the Design Repository.**

   Based on the model dependency graphs, interrogation of the Repository becomes a task of comparing the model dependency graph of a query part to those in the database. The specific comparison that we are interested in is that of the largest common subgraph. The general problem of determining the largest common subgraph for a given pair of graphs is NP-complete [14]. However, in the context of our problem, we observe:

   - First, it is not necessary to find the largest common subgraph: Since we are only concerned with similarity, it is simply necessary to find a "sufficiently" large common subgraph of two MDG's to determine their similarity. Hence, we can use an iterative improvement algorithms for the largest common subgraph computation. Specifically, an iterative improvement search algorithm (a variant on hill-climbing/gradient descent search [36]) that exploits the feature information in the extracted machining features.
   - Second, there is a great deal of domain knowledge present in the CAD model and in the machining features that can reduce the search space. For example, we will only consider mappings that compare similar feature types (i.e., holes map to holes, not to pockets). Additional constraints about vertex degree and size, location, and orientation can also be considered.

We exploit this knowledge to create tractable methods for manufacturing similarity comparisons among solid models.

term of the AlliedSignal Corporation, Federal Manufacturing Systems, presently part of Honeywell (soon to be part of General Electric Corporation).

This paper presents our algorithms as well as empirical experiments based on Allied Signal's FBMach Machining Feature Recognition Husk on a set of 259 CAD models from the National Design Repository. The primary contribution of this research is our overall approach to performing machining similarity comparisons among manufactured artifacts found in modern engineering databases. Ours is the first fully automated technique for machining feature-based comparisons of mechanical artifacts. To achieve this end, this work has created novel data structures, heuristic search and graph comparison techniques that can be applied to a number of important problems in engineering databases and manufacturing process planning. We believe that our work creates part of a foundation that will advance our abilities to manage digital data in distributed engineering enterprises.

2. Background and Related Work

In engineering practice, indexing of parts and part families had been done with group technology coding [39]. Group technology facilitated process planning and cell-based manufacturing by imposing a classification scheme on individual machined parts. GT codes specified classes using alphanumeric strings. These techniques were developed prior to the advent of inexpensive computer technology, hence they are not rigorously defined and are intended for human, not machine, interpretation. At one level, we see our research as augmenting traditional group technology coding schemes by providing a completely digital process for storage and comparison of solid models. It should be noted, however, that at other levels the approach we advocate is not limited to categorization of solid models of machined parts. In recent experiments, we have begun to study how our graph-based approaches can be used to index assemblies, design process knowledge, and CAD data from other domains (e.g., AEC).

2.1. Machining Feature Recognition

Primarily in the areas of manufacturing process planning and solid modeling, past research efforts have developed a variety of techniques for reasoning about geometric and topological information. Much research has been done in the area of automatic feature recognition from three-dimensional solid models [22, 33, 24, 25, 46]. Marefat et al. [24, 27, 26] introduced a novel way of integrating evidence-based reasoning with geometry for process and inspection planning. Marefat's recent work [3] has focused on how to more effectively adapt new planning techniques to process planning. Vandenbrande, Han, and Requichia [45, 46, 21, 22] integrated knowledge-based systems with solid modeling to identify machining features and perform process planning for machined parts. Some of the author's past work on geometric reasoning for manufacturing feature identification and process planning includes [34, 33, 16].

2.2. Comparisons of Shape and Solid Models

The literature in this area is rather brief, consisting of results from engineering, computer science and, in particular, computer vision communities. Elinson et al. [13] used feature-based reasoning for retrieval of solid models for use in variant process planning. Cicirello and Regli [30, 6, 7] examined how to develop graph-based data structures and create heuristic similarity measures among artifacts—work which this paper extends to manufacturing feature-based similarity measurement. Other recent work from the Engineering community includes techniques for automatic detection of part families [29] and topological similarity assessment of polyhedra [41].

The mainstream computer vision research has typically viewed shape matching in approximate domains, such as from models generated from range and sensor data. Work at the University of Utah [42, 12, 43] enables reverse engineering of designs by generating surface and machining feature information off of range data collected from machined parts. Jain et al. and Virage Inc. [15] have been working with Informix to build DataBlades for handling multimedia data such as pictures (GIF, JPEG, etc.) and, more recently, CAD data. Their approach is based on the creation of "feature vectors" from 2D images that capture concepts such as color, density, and intensity patterns. Their work in extending these techniques to 3D CAD data treats the CAD information as sets of point clouds (such as generated with range data) to be compared. While we believe these techniques hold promise, they fail to exploit the availability of 3D solid models representing the CAD data, as well as the engineering information included with CAD data about tolerances, design/manufacturing features, and inter-part relationships that occur in assemblies.

One example where CAD data is employed is the 3D Base System [11, 10] from Dartmouth College. 3D Base operates by converting CAD models (a solid model or surface model) into an IGES-based neutral exchange file. A voxel (3D grid) representation is generated from the IGES data and then used to perform pair-wise comparisons among the CAD files using geometric moments of the voxels and by comparing other per-computed part features (such as surface area). Their work operates only on the gross-shapes of single parts and does not operate directly on the solid models. It does not consider information pertaining to manufacturing or design features, tolerances, or design knowledge that might be present in the corporate database; the voxelization approach would be impractical to scale to electromechanical assemblies, where inter-part relationships and models of function and behavior are much more significant than gross shape properties.

Many other techniques find their roots in computer vision and computer graphics [2, 40, 9]. These, while powerful, are not suitable for off-the-shelf use in similarity matching of solid models.

3. Technical Approach

3.1. Machining Feature Recognition

While our technical approach involves automatic recognition of machining features, this paper is not presenting specific new advances in feature recognition. The central contribution of this work is the methodology for machining similarity comparison of solid models, which is independent of which feature recognizer and which set of machining features one wishes to compare against. We impose three requirements on the feature recognition module:

1. **Recognizes Machining Features:** Recognizers that return shape or form features are not of significant use in assessing machining similarity among artifacts. As has been noted previously [18], machining features contain manufacturing process knowledge in addition to shape information.

2. **Recognizes Interacting Features:** Feature interactions significantly influence the process plans and selection of machining operations and fixtures [35]. Further, all but the most trivial of machined parts have interacting features.

3. **Potential to Return Multiple Feature Interpretations:** In some cases it might be useful to consider the *feature space* of available machining operations in order to assess the properties of an artifact. The problem of multiple interpretations has been widely studied [19] and greatly influences selection of process plans. Using *feature cover* (i.e., the total set of possible machining features) to create the index allows comparisons among artifacts to broaden the consideration to include parts with process plans that come from similar *feature spaces*.

There are many academic and research prototype feature identification systems. For a recent survey on machining feature recognition from solid models, interested readers are referred to [23]. In this research, we chose to use one of the few available industrial systems: the Feature-Based Machining Husk (FBMach) [4, 5] from AlliedSignal, Federal Manufacturing Technologies in Kansas City. FBMach is a robust library of machining features and feature recognition algorithms. It comprises approximately 10 man-years of effort and several hundreds of thousands of lines of code.

FBMach uses three different approaches to define surface features: (1) automatic recognition, (2) interactive recognition and (3) manual identification. The automatic recognition uses a procedural algorithm to search for feature hints and then creates feature instances using the hints without user interaction. The interactive recognition allows the user to provide some hints for FBMach to use in generating the feature instances. For example, the user may identify a pocket by selecting its bottom face. The manual identification allows the user to create a feature instance by adding each face to the feature individually and defining each face's role in the feature (side, bottom, top, etc.). FBMach implements a *human-supervised reasoning* approach, which has also been explored by van Houten [44] along a different direction. Such a human-supervised reasoning is often quite useful for producing good feature models.

We used a slightly modified version of FBMach that translated unattributed ACIS `.sat`-based solid models into sets of STEP AP 224 [38] NC machining feature volumes. The feature sets returned by FBMach were used to create the *Model Dependency Graphs* described in the next Section and were the basis of the empirical results described in Section 4.

3.2. Definition and Generation of Model Dependency Graphs

A **Model Dependency Graph** (MDG) is defined in [6] as a representation of the design features and interdependencies of those design features of a CAD model. This graph is a directed acyclic graph and has some unique characteristics. The nodes of this graph correspond to individual design features. An edge between two nodes corresponds to some spatial dependence between the features (i.e. à non-empty intersection of the feature volumes). The direction on the edges capture the order that design features were applied during the design phase.

Observation: *D-morphisms of Model Dependency Graphs.* Let G_1 and G_2 be two MDGs for the same solid model resulting from different orderings of a feature set $F = \{f_0, \dots, f_n\}$ (such as shown in Figure 1). G_1 and G_2 are D-morphic. A proof of this property appeared previously in [6].

For a given pair of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ a *D-morphism* is formally defined in [14] as a function $f : V_1 \to V_2$ such that for all $(u, v) \in E_1$ either $(f(u), f(v)) \in E_2$ or $(f(v), f(u)) \in E_2$ and such that for all $u \in V_1$ and $v' \in V_2$ if $(f(u), v') \in E_2$ then there exists a $v \in f^{-1}(v')$ for which $(u, v) \in E_1$.

As defined above, the model dependency graph is not immediately applicable to machining features. If we knew the order in which the machining operations were performed, then perhaps it would be directly applicable. In this case, the D-morphism property would also apply. In this paper we are considering machining features that are extracted post facto. We do not know what order they were performed and will not attempt to designate some arbitrary and meaningless ordering on the machine features. Rather we will exploit an undirected version of the MDG which we shall call the **Undirected Model Dependency Graph**.

Undirected Model Dependency Graph: The undirected model dependency graph (UMDG) $G = (V, E)$ for a solid model is defined as a set of nodes $G = \{f_0, \dots, f_n\}$ where the f_i are the machining features that have been extracted from the model. The edge set E of the UMDG can be defined as: $E = \{\{f_i, f_j\}\}$ such that $vol(f_i) \cap vol(f_j) \neq \emptyset\}$.

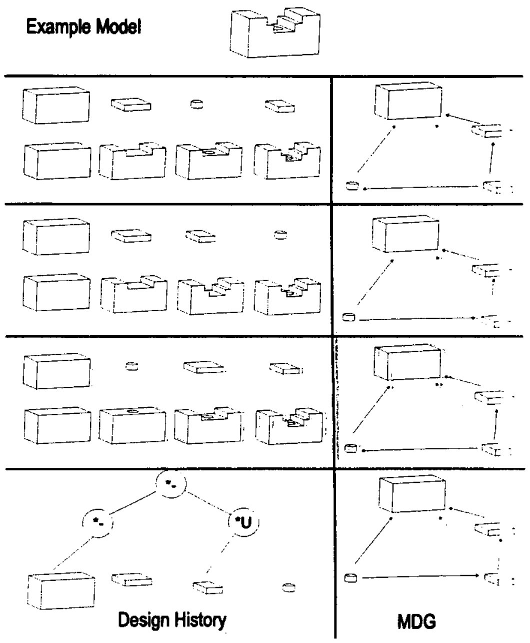


Figure 1. (From [6]) Pictured is a single solid model and several alternative feature-based models, and one possible CSG tree, that can produce it. On the right are the MDGs for each of these alternatives—note that they are all D-morphic to one another.

3.3. Part Retrieval

To compare the similarity of two solid models, compute the size of the largest common subgraph (LCS) of the corresponding UMDGs. The largest common subgraph problem is to find a pair of subgraphs, one from each of the two input graphs, such that the subgraphs in question are isomorphic to each other and the largest such subgraphs in terms of the number of edges. As noted earlier, the largest common subgraph problem (LCS) is NP-complete and the existence of an algorithm computable in polynomial time is not likely to exist.

We are primarily concerned with the manufacturing feature similarity among artifacts. Hence, knowing the absolute largest common subgraph of the UMDGs is not necessary. Instead, a "large enough" common subgraph is sought. There exist some inexact solutions in the literature including an approach using a two-stage Hopfield neural network [37] and a meta-optimized genetic algorithm [8]. There are also some inexact solutions to the very closely related problem of error-correcting isomorphism including a decision tree approach [28], a linear programming approach [1], and a genetic algorithm [47].

We develop an heuristic measure method for the computation of the LCS based on a variant of iterative improvement search (specifically, hill-climbing/gradient descent) [36]. To further refine and narrow the search space, our algorithm utilizes the domain knowledge present in the CAD model. For example, we will only consider mappings that compare similar feature types (i.e., holes map to holes, not to pockets). Additional constraints about vertex degree and size, location, and orientation can also be considered.

The following sections present our approach. While this is not guaranteed to find an isomorphism if one exists, it allows for a *measure of manufacturing similarity* based on the best result obtained from executing some number of restarts of the algorithm.

3.4. MDG Approximate Matching Algorithm

In searching for the LCS, first, arbitrarily choose an initial mapping between the nodes of the two graphs (i.e., for each node of G_1 choose at random a node of G_2 such that no two nodes of G_1 are mapped to the same node of G_2). Next, swap the mappings of the two nodes that reduce the value of the evaluation function the most. If there is no swap that reduces the value of the evaluation function, but there are swaps that result in the same value (i.e., a plateau has been reached), choose one of those at random. The algorithm ends when either every possible swap increases the value of the evaluation function or it makes P random moves on the plateau. Values of P ranging from constant values to $P = |V_1|^2$ (where V_1 is the vertex set in the smaller graph) have been experimented with.

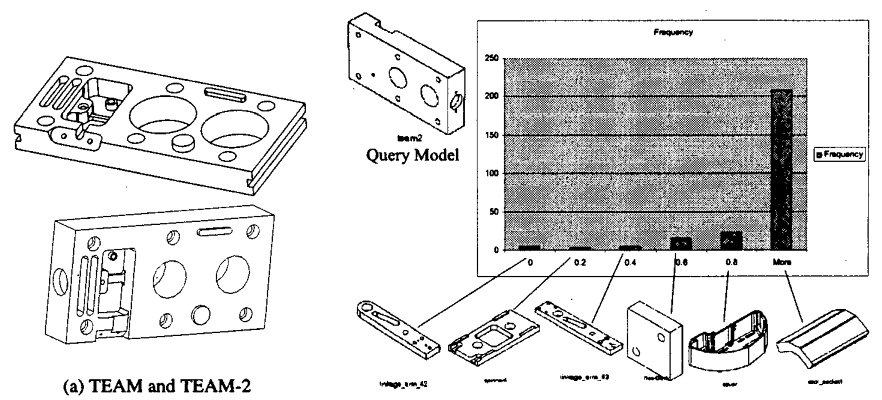
The evaluation function is the count of the number of mismatched edges. That is, the evaluation function, $H = |E|$ such that $G_1 = (V_1, E_1)$ is the smaller of the two graphs being compared, $G_2 = (V_2, E_2)$ is the larger of the two graphs, and $E = \{(u, v) \in E_1$ such that $(((paired(u), paired(v)) \notin E_2 \wedge (paired(v), paired(u)) \notin E_2)) \vee label(u) \neq label(paired(u)) \vee label(v) \neq label(paired(v))\}$. As a measure of similarity employ the value $H^* = \frac{min\{H_1,\ldots,H_n\}}{|E_1|}$ where $H_1, \ldots, H_n$ are the final values of H from up to n random restarts of the algorithm and E_1 is the edge set of the smaller graph. The function "paired(x)" above returns the node $y \in V_2$ that is currently mapped to the node $x \in V_1$. The function "*label(x)*" used above returns the label, or attributes, of the node x.

Algorithm 1: Largest Common Subgraph
Input: $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$, the two graphs being tested. P is the number of moves to make on a plateau before giving up.
Output: $H = 0$ if the LCS is found to correspond to a subgraph isomorphism. Otherwise, H is returned where H is the number of mismatched edges when the algorithm halts.
LCSGRADIENTDESCENT(G_1, G_2, P)

```
(1)  Pairings = GETRANDOMPAIRINGS($G_1, G_2$)
(2)  $i = 0$
(3)  BestResult = $H(G_1, G_2$,Pairings)
(4)  while (BestResult > 0) $\wedge$ ($i < P$)
(5)      if $H(G_1, G_2$,APPLYSWAP(Pairings,BestSwap)) <
         BestResult
(7)          Pairings = APPLYSWAP(Pairings,BestSwap)
(8)          $i = 0$
(9)          BestResult = $H(G_1, G_2$, Pairings)
(10)     else
(11)         if $H(G_1, G_2$, APPLYSWAP(Pairings,BestSwap)) =
             BestResult
(12)             Pairings =
                 APPLYSWAP(Pairings, BestSwap)
(14)             $i = i + 1$
(15)         else
(16)             $i = P$
(17) return BestResult
```

The node labels may contain as little or as much information as you choose. For the experiments that are described later, the node labels were simply the type of feature, such as "hole" or "pocket". However, by incorporating more information into the node labels such as dimensions or orientation, we can further restrict allowable mappings which will increase the algorithm's performance by reducing the search space. Incorporating more information in the node labels will also obtain a more meaningful similarity measure. For example, by incorporating a notion of dimension into the labels then a really large block with a tiny hole will not be found similar to a little block with a larger hole.

Algorithm 1 is the algorithm developed and described for computing the largest common subgraph using iterative improvement. In the algorithm, Pairings refers to the mapping between the nodes of the two graphs. And GETRANDOMPAIRINGS returns a random mapping as de-



(a) TEAM and TEAM-2

(b) Query Results

Figure 2. Two of the test parts from the DOE TEAM Project. Note that, while they appear very similar, they have variations in features, geometry and topology—as well as orientation with respect to the world coordinate system. Both of these parts are available from the National Design Repository at `http://www.designrepository.org`.

Algorithm 2: Similarity
Input: $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$, the two graphs being compared. R is the number of restarts.
Output: $S = 0$ if the smaller of the two graphs is the largest common subgraph. Otherwise, S is returned where S is the smallest result of R restarts of LCSGradientDescent divided by the number of edges in the smaller of the two input graphs.

```
SIMILARITY($G_1, G_2, R$)
(1) $i = 1$
(2) BestResultThusFar = LCSGRADIENTDESCENT($G_1, G_2, P$)
(3) while (BestResultThusFar > 0) $\wedge$ ($i < R$)
(4)     BestResultThusFar = min {BestResultThusFar,
        LCSGRADIENTDESCENT($G_1, G_2, P$) }
(5)     $i = i + 1$
(6) return $\frac{BestResultThusFar}{\min\{|E_1|,|E_2|\}}$
```

scribed above. H is the evaluation function that counts the number of mismatched edges given two graphs and a mapping between the nodes in these two graphs. `BestSwap` is the swap from the set of all possible swaps between pairings that results in a mapping with the smallest value for H. APPLYSWAP returns the mapping that results from applying the given swap to the given mapping. The algorithm is of polynomial time complexity. It takes $O(N^2)$ time to choose the best swap. In the worst possible case, by choosing the best swap at each step the evaluation function is simply reduced by one and therefore can look for the best swap as many as $|E|$ times. It takes time in $O(|E|)$ to compute the evaluation function. Also in this worst case, the algorithm reaches a plateau as often as possible and takes P random moves on each of these plateau before finding the swap that reduces the evaluation function. So therefore the worst case complexity of the algorithm is $O(P*E^2 + P*E*N^2)$. If P is a constant then the complexity is simply $O(E^2 + E*N^2)$. To obtain a similarity measure, the smallest result of r executions of this algorithm is divided by the number of edges in the smaller of the graphs. Algorithm 2 is the random restart algorithm for similarity assessment. The similarity algorithm simply calls the matching algorithm r times. Since r is constant the complexity is $O(E^2 * N^2)$.

4. Empirical Results

We ran the FBMach feature recognizer over a set of 259 solid models for real world and realistic machined parts. The parts were chosen from the National Design Repository and the feature output was used to generate MDGs for each of the models—these MDGs formed the indexing scheme against which we performed queries. The parts were selected based on their diversity and that they were machined parts, many of which were from industry. We conducted several "query by example" experiments, selecting a representative solid model as a target and letting our matcher estimate the distance from the other models in the dataset to the target.

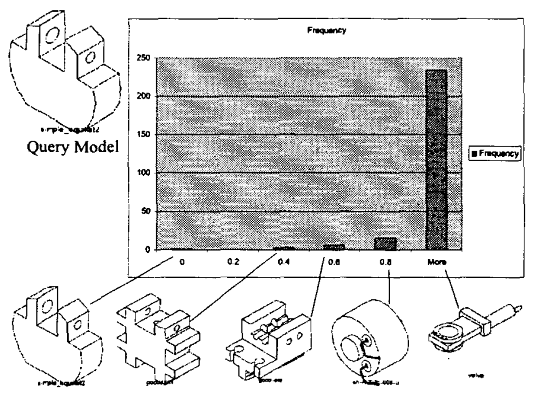


Figure 3. The Simple Bracket example, originally from [17]. When used as a query, the query processor identifies several parts, including variations on the bracket, in the "most similar" category.

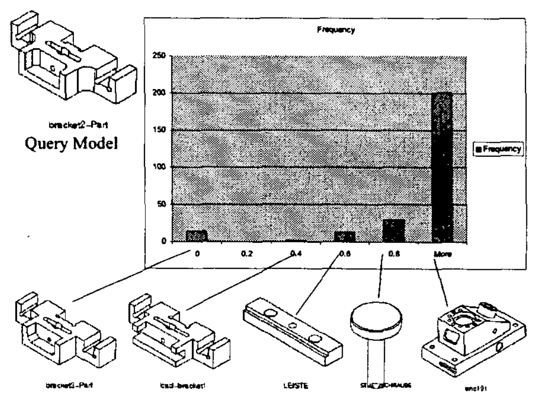


Figure 4. The Bracket example, originally from [16]. Several variations on the bracket appear in the near hits buckets of the histogram.

Testbed Description: The National Design Repository. The National Design Repository [32, 31] is a digital library of Computer-Aided Design (CAD) models and engineering designs from a variety of domains. The objective is to further the state-of-the-art in academic and industrial research in Computer-Aided Engineering by building a public catalog of real-world design examples. The Repository provides benchmark designs in a variety of formats to serve as reference data to aide developers and students.

The Repository currently contains over 55,000 files maintained in multiple data file formats[2]. Contributions have been made by many major research laboratories, companies and academic institutions. Currently there are 10 gigabytes of CAD models and related information. All data is freely available for users around the world. More information is available at `http://www.designrepository.org`.

How to interpret the histograms and model matching results. Figures 2 to 5 give histograms of the results of running a comparison of the query model (shown in the upper-left of each Figure) against the MDGs of all of the other models indexed by FBMach in the Design Repository. Reading the histograms from left-to-right, the parts on the left are more similar to the query and the parts on in the right most buckets are the least similar from a machining standpoint.

The distance between solid models is measured as the minimal percentage of mismatched edges as calculated over the course of several runs of the LCS approximate MDG matcher. Referring to the histograms, the vertical axis (Y axis) is a count of the number of solid models falling into each one of the six buckets; the horizontal X axis is the percentage of mismatched MDG edges (i.e., the ratio of mismatched edges to total number of edges in the smaller of the two MDGs). For example, referring to Figure 2 (c), there were 5 models whose MDG's exactly matched (or were embeddable in) that of the query model; 3 models with 20% or fewer mismatched edges; 5 models with 21%-to-40% mismatched edges; 15 with 41%-60% mismatches; 23 models with 61%-to-80%; and 208 models with greater than 80% mismatches. In this way, the histograms show a partition of the parts into groups based on the estimated distance between their MDG and the MDG of the query object.

The comparisons, as noted in Section 3, amounts to a subgraph isomorphism check and distance measure, the CPU time needed to execute these checks were quite reasonable—often running in real time. We hypothesize this performance is achieved because we are operating on labeled graphs and that the iterative improvement approach allows us to control the duration of the search. In this class of experiments, we ran the search with 10 random restarts and took the best matches off of the 10 trials. Specific CPU times from a Sun UltraSPARC 30 workstation are noted below with each example.

Four Example Queries. Four (4) of the parts in the Repository were selected as query models. Their selection was based on the knowledge that, in each case, there were

[2]This includes solid models in STEP AP 203, ACIS `.sat`, Autodesk `.dxf` and `.dwg`, IGES, Bentley `.dgn`, Parasolid `.xmt` and other formats

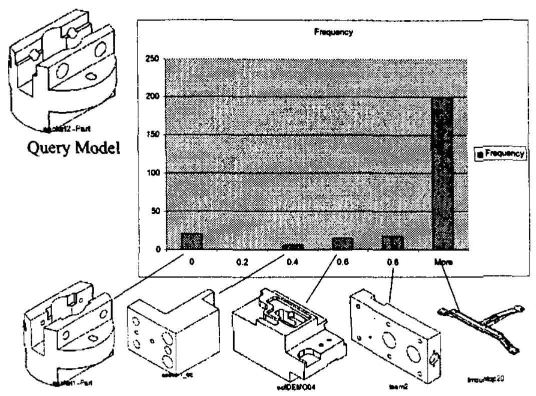


Figure 5. The Socket example, originally from [33]. A variations on the socket appears in the nearest hit bucket.

models existing in the Repository which indeed would have similar process plans.

TEAM Part: Figures 2 (a) and (b) show the two TEAM Parts, from the US Department of Energy's Technologies Enabling Agile Manufacturing (TEAM) Project. Note that they are both three axis machined parts with minor variations in features, geometry and topology. When TEAM2 was used as a query (Figure 2 (b)), TEAM (Figure 2 (a)) and several other three axis machined parts with similar features and topology appear in the near hit category. For this query, it took 304.71 CPU seconds to search across all 259 models.

Simple Bracket: Figure 3 shows a simple bracket, a part with about a dozen possible machining feature that can be made in three or four set ups (depending on tolerance constraints). The histogram shows the result of comparing the features in this model to those of the others in our Repository. For this query, it took 5.4 CPU seconds to search across all 259 models.

Bracket: Figure 4 shows a more complex bracket structure. The histogram of results from this query show two variations on the bracket appearing in the left-most classes of models. Note that these variations change the setups and the feature types (drilled holes to machined slots) but have similar shape properties. Further note that the vast majority of models fall in to the least similar categories. For this query, it took 21.27 CPU seconds to search across all 259 models.

Socket: Figure 5 shows the Socket example, consisting of 22 machining feature instances and machinable in four setups. Note that a near duplicate of the socket appears in the most similar category. For this query, it took 95.44 CPU seconds to search across all 259 models.

5. Discussion and Conclusions

This paper presented our approach to using machining features as an index-retrieval mechanism for storing solid models. We believe that this work represents the the first fully automated technique for machining feature-based comparisons of mechanical artifacts. Our hope is that this type of approach, combining raw b-Rep data with machining feature knowledge, will enable to radical changes in the way in which design data is managed in modern engineering enterprises.

Our approach enables us to interrogate large databases of solid models and perform queries based on manufacturing similarities among the artifacts. While we have not yet performed a comprehensive analysis of the manufacturing data for each of the parts in our Design Repository, our empirical results suggest that this is a promising approach to information and data management for design and manufacturing process knowledge.

Research Contributions and Future Work. Some of the research contributions of this work include:

A Model Indexing and Query Scheme: We showed that the MDG is a useful mechanism for archival and retrieval of models in CAD databases and can be employed using a "query by example" paradigm. Using algorithms for computing the largest common subgraph, and introducing some engineering domain knowledge, we have created a general technique for archiving large numbers of solid models and retrieving them based on the similarity of their machining features. We believe that this technique can be refined and will have impact on how CAD data is stored and managed.

Variational Process Planning Search Engine: Based on the MDG, one can create query artifacts that partition the database of solid models into different morphism classes—based on how similar in structure each model is to the query model. We believe that this approach can be refined to detect meaningful part classes and families in large sets of engineering models. This can form the basis for more intelligent Product Data Management (PDM) systems and tools for variational design and variant process planning.

Process Selection and Cost Estimation Engine: Corporate Design and Manufacturing databases often store manufacturing cost and process data.

With the techniques presented, we can design Active CAD Agents that assist designers during detailed design by comparing the in-process design artifact to those previously created. In this way, the agent can provide feedback on potential manufacturing process choices (e.g., when a design that is categorized as similar was made with stereolithography, perhaps stereolithography should be considered for the new design) and expected cost (e.g., using data from the existing process plan for similar parts).

Currently, our research considers only plain, unattributed solid models—there are no tolerances, manufacturing attributes, surface finish specifications, etc. In the future, we believe that additional domain knowledge can be used to refine our techniques. Information about engineering tolerances, surface finishes, constraints and parametrics, etc. all can be used to augment the basic techniques presented here. Further, we would like to explore how to implement multiple feature views onto the Repository by including feature data generated by different feature recognition techniques.

Acknowledgements. Our thanks are extended to Dr. Steve Brooks of Honeywell, Federal Manufacturing Technologies Program, in Kansas City for providing the National Design Repository with the ACIS models for the TEAM parts. This work was supported in part by National Science Foundation (NSF) Knowledge and Distributed Intelligence in the Information Age (KDI) Initiative Grant CISE/IIS-9873005; CAREER Award CISE/IIS-9733545 and Grant ENG/DMI-9713718. 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 National Science Foundation or the other supporting government and corporate organizations.

References

[1] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):522–525, May 1993.

[2] B. Bhanu. Surface representation and shape matching of 3-d objects. In *PRIP82*, pages 349–354, 1982.

[3] J. Britanik and M. Marefat. Hierarchical plan merging with applications to process planning. In *International Joint Conference on Artificial Intelligence*, 1995.

[4] S. L. Brooks and R. B. Greenway Jr. Using STEP to integrate design features with manufacturing features. In A. A. Busnaina, editor, *ASME Computers in Engineering Conference*, pages 579–586, New York, NY 10017, September 17-20, Boston, MA 1995. ASME.

[5] S. L. Brooks and M. L. Wolf. Overview of Allied Signal's XCUT system. In J. Shah, M. Mäntylä, and D. Nau, editors, *Advances in Feature Based Manufacturing*, pages 399–422. Elsevier/North Holland, 1994.

[6] V. Cicirello and W. C. Regli. Resolving non-uniqueness in design feature histories. In D. Anderson and W. Bronsvoort, editors, *Fifth Symposium on Solid Modeling and Applications*, New York, NY, USA, June 8-11 1999. ACM, ACM Press. Ann Arbor, MI.

[7] V. A. Cicirello. Intelligent retrieval of solid models. Master's thesis, Drexel University, Geometric and Intelligent Computing Laboratory, Department of Mathematics and Computer Science, Philadelphia, PA 19104, June 1999 1999.

[8] V. A. Cicirello and S. F. Smith. Modeling GA performance for control parameter optimization. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, July 2000.

[9] E. Cohen and H. Wolfson. Partial matching of 3-d objects in cad/cam systems. In *ICPR94*, pages A:483–487, 1994.

[10] K. D. Cohen. Feature extraction and pattern analysis of three-dimensional objects. Master's thesis, Dartmouth College, Thayer School of Engineering, Hanover, NH, 1996.

[11] G. Cybenko, A. Bhasin, and K. D. Cohen. 3d base: An agent-based 3d object recognition system. http://comp-eng-www.dartmouth.edu/~3d, January 1996.

[12] H. J. de St. Germain, S. R. Stark, W. B. Thompson, and T. C. Henderson. Constraint optimization and feature-based model construction for reverse engineering. In *Proceedings of the ARPA Image Understanding Workshop*, Feburary 1996.

[13] A. Elinson, D. S. Nau, and W. C. Regli. Feature-based similarity assessment of solid models. In C. Hoffman and W. Bronsvoort, editors, *Fourth Symposium on Solid Modeling and Applications*, pages 297–310, New York, NY, USA, May 14-16 1997. ACM, ACM Press. Atlanta, GA.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 41 Madison Avenue, New York, NY 10010, 1979.

[15] A. Gupta and R. Jain. Visual information retrieval. *Communications of the ACM*, 40(5):71–79, May 1997.

[16] S. K. Gupta, T. R. Kramer, D. S. Nau, W. C. Regli, and G. Zhang. Building MRSEV models for CAM applications. *Advances in Engineering Software*, 20(2/3):121–139, 1994. *Special issue on Feature-Based Design and Manufacturing*.

[17] S. K. Gupta and D. S. Nau. A systematic approach for analyzing the manufacturability of machined parts. *International Journal of Computer Aided Design*, 27(5):323–342, 1995.

[18] S. K. Gupta, W. C. Regli, and D. S. Nau. Manufacturing feature instances: Which ones to recognize? In J. Rossignac, J. Turner, and G. Allen, editors, *Third Symposium on Solid Modeling Foundations and CAD/CAM 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.

[19] J. Han. On multiple interpretations. In *4th ACM SIGGRAPH Symposium on Solid Modeling and Applications*, pages 311–321. ACM Press, May 14-16 1997. Atlanta, Georgia.