Achieving Reconfigurable CAD/CAPP Integration by Reasoning from Shape Producing Capabilities

Daniel M. Gaines
dmgaines@cs.uiuc.edu
Department of Computer Science

Fernando Castaño
castano@students.uiuc.edu
Department of
Electrical and Computer Engineering

Caroline C. Hayes
hayes@cs.uiuc.edu
Department of Computer Science

University of Illinois at Urbana-Champaign

Abstract

This paper presents MEDIATOR, a feature recognition system which is designed to be maintainable and extensible to families of related manufacturing processes. A problem in many feature recognition systems is that they are difficult to maintain. One of the reasons may be because they depend on use of a library of feature-types which are difficult to update when the manufacturing processes change due to changes in the manufacturing equipment. The approach taken by MEDIATOR is based on the idea that the properties of the manufacturing equipment are what enable manufacturable shapes to be produced in a part. MEDIATOR's method for identifying features uses a description of the manufacturing equipment to simultaneously identify manufacturable volumes (i.e. features) and methods for manufacturing those volumes. Maintenance of the system is simplified because only the description of the equipment needs to be updated in order to update the features identified by the system.

Introduction

This paper presents MEDIATOR (Maintainable, Extensible Design and manufacturing Integration Architecture and TranslatOR). The goal of MEDIATOR is to provide a maintainable and extendible feature recognition method for linking computer-aided design (CAD) systems to computer-aided process planning (CAPP) systems. In this paper we will mainly address the operation of MEDIATOR in the domain of feature recognition for 3-axis machining centers and briefly discuss how MEDIATOR can be adapted to other domains. MEDIATOR's approach is to derive feature descriptions directly from descriptions of the manufacturing equipment.

MEDIATOR's current scope is as a feature recognizer and method generator for 3-axis universal machining centers. Even within the domain of 3-axis milling, it is useful to have a feature recognizer that can be adapted to different shops having different tools. MEDIATOR can be easily reconfigured to reflect the resources f a given shop. Different shapes may be manufacturable in different shops because of differences in the available tools, even if very similar manufacturing processes are used. In the near future, we hope to extend MEDIATOR to be reconfigurable for a family of related material removal processes, such as 3 and 5 axis milling, turning, and mill-turning. Additionally, we feel that MEDIATOR's general approach can be used to construct maintainable and extendible feature recognizers for various manufacturing processes.

Many approaches to feature recognition use a library containing descriptions of classes of task-relevant features, (such as holes, pockets and slots). Specific features are identified in a part description by searching the part for groupings of geometry matching the feature descriptions. Other approaches use context-independent form-features, however, in order to convert the task-independent features into a form that will be useful for specific domain tasks, it is necessary to use a task-specific post processor. The drawback of such approaches is that the task-specific parts are hard to update and maintain. Changes in the tools and equipment may result in changes to the set of shapes that can be manufactured. However, it is a difficult task for a system maintainer to figure out what corresponding changes need to be made to the feature descriptions in the library.

We feel that part of this difficulty may arise from the fact that feature libraries and form features may not be the most natural or effective way to represent the link between design and manufacturing. We feel that a more natural approach, which we have implemented in MEDIATOR, is to reason from knowledge about manufacturing equipment to simultaneously identify manufacturable areas of the part, and a manufacturing method which can produce them. A *manufacturing method* is a specification of a tool-type, fixture-type, and part orientation which will be used to manufacture a particular volume. MEDIATOR can be maintained by making changes to the equipment descriptions.

Some unusual properties resulting from this view of

feature recognition are that fixture information is used in feature recognition, and that feature volumes and methods for cutting those volumes are generated simultaneously. The implication is that feature recognition is tightly tied to process planning, and that the boundary between the two is somewhat blurred.

Related Work

This section describes relevant work in feature recognition for CAD/CAPP integration. We divide previous feature recognition research into two main categories of algorithms, those based on feature-class descriptions and those based on form-features. Also included is work related to fixture analysis.

Feature-class approaches. (Choi, Barash, & C. 1984, Joshi & Chang 1988, Marefat & Kashyap 1990, Vandenbrande & Requicha 1993, Gupta *et al.* 1994, Regli, Gupta, & Nau 1994) use a library of features to characterize the shapes that are needed to perform analysis with respect to some task such as machining or assembly. The feature recognizer searches the part model for geometry matching these feature descriptions. A drawback to these approaches is that feature-class libraries are difficult to maintain and extend. While it is possible to apply these techniques to different domains it is difficult to 1) ensure that the feature-classes adequately cover the set of shapes which can be produced in that domain, and 2) maintain the system as that domain changes. Changes to the properties of the target task may require extensive modifications to the feature-class libraries.

Form-feature approaches. Form-feature approaches attempt to achieve more generality by avoiding the use of task-specific feature descriptions (Woo 1982, Kim 1994, Shah, Shen, & Shirur 1994). A difficulty with these approaches is that because form-features are independent of particular tasks and context, much post-processing work may be required to translate them into task-specific features which can be used for tasks such as milling, turning or assembly. For example, (Shah, Shen, & Shirur 1994, Gaines, Hayes, & Kim 1995) map form-features to machining methods. Essentially, these approaches do not avoid the maintenance problem, but only push it into the task-specific post-processor. Additionally, they do not allow task-specific knowledge to help direct the decomposition of the part model.

MEDIATOR does not make use of a set of feature-class libraries, nor does it perform a mapping from form-features to manufacturing methods. Instead, knowledge about the equipment, such as tool shapes, tool dimensions, part and tool motions, part orientations, etc., is used to infer which areas of the part can be made with the equipment. In the process, features are built up along with the methods that can be used to produce them. Other distinctions from the work of (Shah, Shen, & Shirur 1994, Gaines, Hayes, & Kim 1995) include the feature recognition algorithm and the structure of equipment knowledge. The scope of equipment knowledge includes fixture, process (tool motion), and tool knowledge rather than just tool and process information.

Fixture analysis. Since MEDIATOR uses some fixturing analysis in performing feature recognition we will also discuss some fixture work. In particular we will focus on process planners that incorporate fixture analysis (Chang & Anderson 1992, Cutkosky & Tenenbaum 1992, Nau, Gupta, & Regli 1995, Das, Gupta, & Nau 1994). The difference between these works and MEDIATOR is that they bring in fixturing analysis only in the process planning stage but not as early as feature recognition.

The following section described MEDIATOR's feature recognition and method generation process.

MEDIATOR: A Reconfigurable Feature Recognizer

This section describes the MEDIATOR feature recognition system. Figure 1 shows the architecture of MEDIATOR. MEDIATOR takes as input the boundary representation of a part model and the initial stock from the CAD modeler. MEDIATOR simultaneously identifies features, and generates one or more manufacturing methods to produce each feature. A feature is recognized as a feature specifically because a method could be constructed to produce the feature. The first step is to generate the *delta volume*, which is all the material to be removed from the stock. It is obtained by subtracting the part from the stock. Next, feature fragments are created from each part face in the delta volume. These feature fragments are joined into larger features through an iterative process called Feature Identification and Method Generation.

The following sections describe the components of MEDIATOR in more detail. The discussion will use the part shown in Figure 2 for illustration. Results for a more complex example are provided in Section .

Generate delta volume

MEDIATOR begins by generating the delta volume by subtracting the part from the stock. The delta volume for the example part is shown in Figure 3. As Figure 3 shows MEDIATOR is capable of handling near-net shape stock.

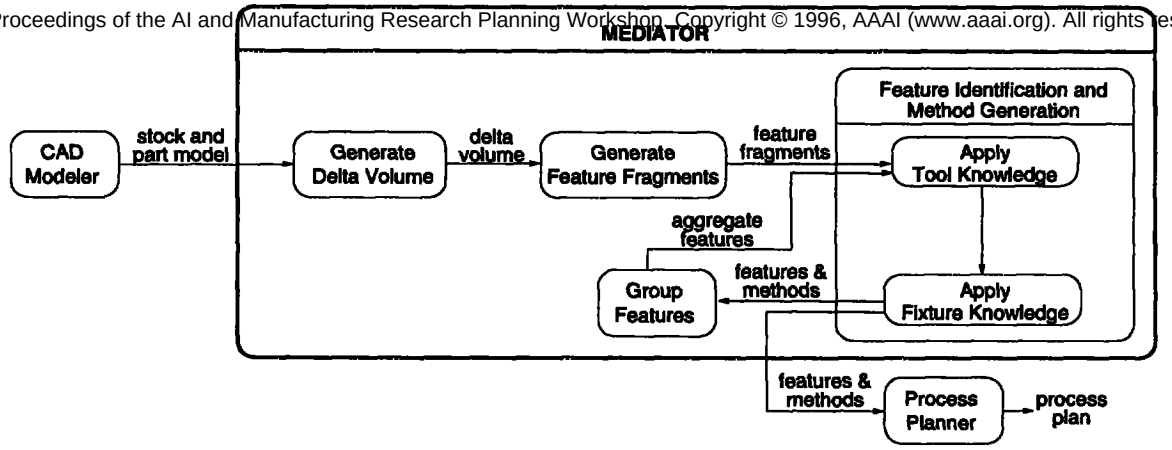


Figure 1: MEDIATOR system diagram.

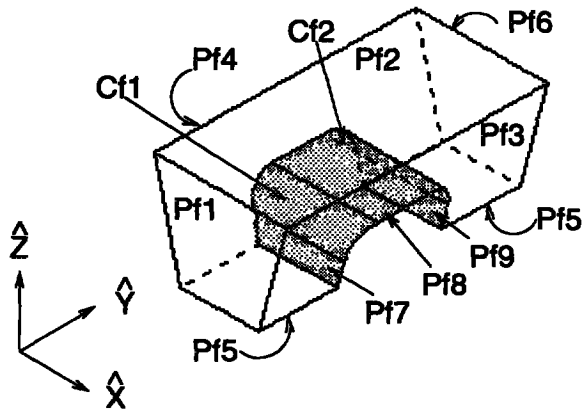


Figure 2: Part I.

Generate Feature Fragments

For each part face in the delta volume, a *feature fragment* is created. These act as "seeds" to the feature recognition process. Larger feature descriptions and manufacturing methods will be built up from these fragments. For the delta volume shown in Figure 3 there are 5 feature fragments, shown as shaded surfaces in Figure 2: Pf7, Pf8, Pf9, Cf1 and Cf2.

Feature identification and method generation

Feature identification and method generation has two main stages: Apply Tool Knowledge, and Apply Fixture Knowledge. During these stages, the feature fragments are refined to create features while outlines for methods for producing them are generated. The remainder of this subsection will follow the progress

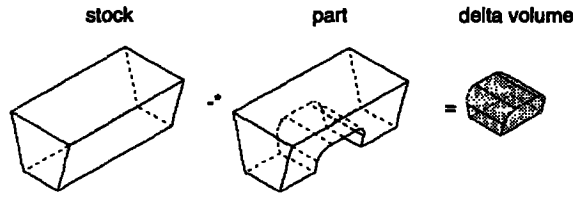


Figure 3: Delta volume for example part I.

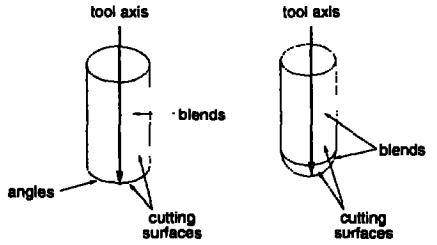
of feature fragment Pf7 as it proceeds through these stages. At each step, the evolving feature and method description will be shown.

Apply tool knowledge At this stage of feature recognition and method generation, tool knowledge is used to construct part of the method description for the feature fragments. There are three substages to this process: static tool requirements, tool motion requirements and volume generation.

Static tool knowledge. Figure 4 shows the knowledge hierarchy representing these tools. It contains knowledge about the static properties of tools which are important in producing shapes: tool shape and individual tool dimensions. Each of these properties has its own level in the hierarchy. The dynamic properties of the tools are represented in another hierarchy.

At the *tool shape level* of the hierarchy , each node contains properties common to classes of tools having a similar shape. For example, in the Flat Endmills node, a generic flat endmill shape is stored with labeled cutting surfaces and a tool axis direction, as shown in Figure 5 (a). Figure 5 (b) shows a similar example for the information stored at the Ball Endmills node. The *tool instance level* describes the specific dimensions of

particular cutting tools at a given machine shop. MEDIATOR can be customized for a particular machine shop by filling the hierarchy with the equipment available at a particular site.



(a) flat endmill (b) ball endmill

Figure 5: Cutting tool representation.

Static tool requirements. Static properties of the tool are used to determine the orientations in which the available tools shapes can produce the feature fragments. The ways in which a feature fragment intersects its adjacent faces places constraints on the tool shapes and the possible orientations. For example, if two planar faces intersect at a right angle, this produces the constraint that a ball endmill cannot create that face from any direction.

For each feature fragment, all intersections it makes with the other part faces in the delta volume are examined. For example, the feature fragment corresponding to surface Pf7 intersects only one other face, Cf1, as shown in Figure 6. This face intersection describes what we call a *blending interaction* between surfaces Pf7 and Cf1.

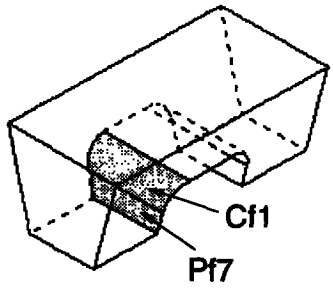
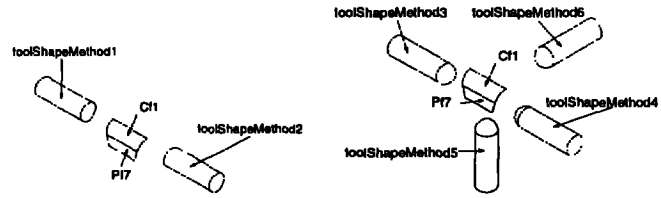


Figure 6: Face Intersection for surface Pf7, type: blend.

The system considers each tool shape and finds the orientations in which it could machine each face intersection. The face intersection between faces Pf7 and Cf1 could be produced with a flat endmill that approaches the part from either the front or the back (Figure 7 (a)). If a ball endmill is used, there are four possible approach directions, as shown in Figure 7 (b). However, one of these, toolShapeMethod6, will be identified as invalid during volume generation since the feature is not accessible from this direction.



(a) flat endmill (b) ball endmill

Figure 7: Tool shape methods for Pf7.

Figure 8 shows the status of the feature and methods for Pf7 after the tool shape knowledge is applied. The feature has grown to include the face intersection with Cf1 (Figure 8 (a). Methods describing the tool shapes and their possible approach directions have been attached to the feature. The set of partial method descriptions that can make this feature are represented in Figure 8 (b) as a tree. There are eight possible methods represented, some of which may be pruned later. The root of the tree shows the name of the feature, the next level shows the possible tool types that can be used, and the last level shows possible approach directions for the tools. In future versions of the system, knowledge about tool instances will be used to further prune the methods.

Tool motion requirements. The feature's shape places requirements on the tool motions which will be needed to produce that shape. This decision is based on the surface and the approach direction of the tool. For example, a convex cylindrical face would only require $\frac{1}{2}$ degree of tool motion freedom if the tool is oriented along the cylinder's axis, whereas a planar surface would require at least $1\frac{1}{2}$ degrees of tool motion freedom. Because Pf7 is a planar face, a cutting tool will require at least $1\frac{1}{2}$ degrees of freedom to cut it.

At this point, some of the methods are found to be invalid since they do not have the motion requirements necessary to produce Pf7. As Figure 9 shows, the methods using a drill have been discarded. The drill was originally considered for this feature since it is able to produce the face intersection between Pf7 and Cf1. However, the drill cannot make the planar face Pf7.

The result of this step was to augment the methods with tool motion requirements, as shown in Figure 9, and to prune methods that were not applicable.

Volume generation. At this stage, *removal volumes*, the volume which will be removed from the workpiece if this feature fragment is valid, are generated, and tool accessibility is checked. Volume generation makes use

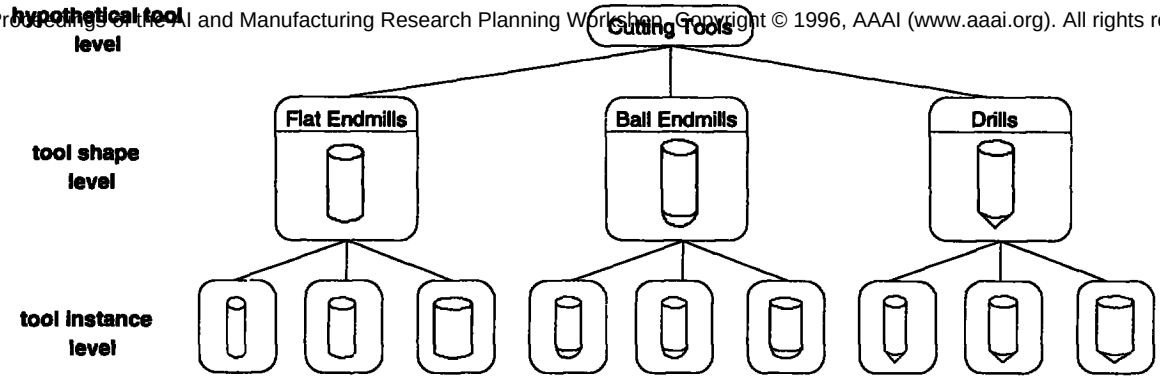
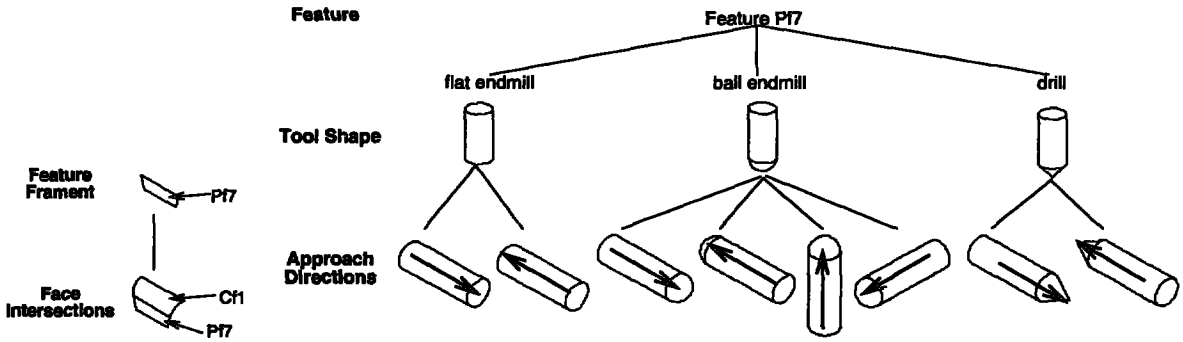


Figure 4: Cutting tool knowledge hierarchy.



(a) Pf7 feature (b) Pf7 methods

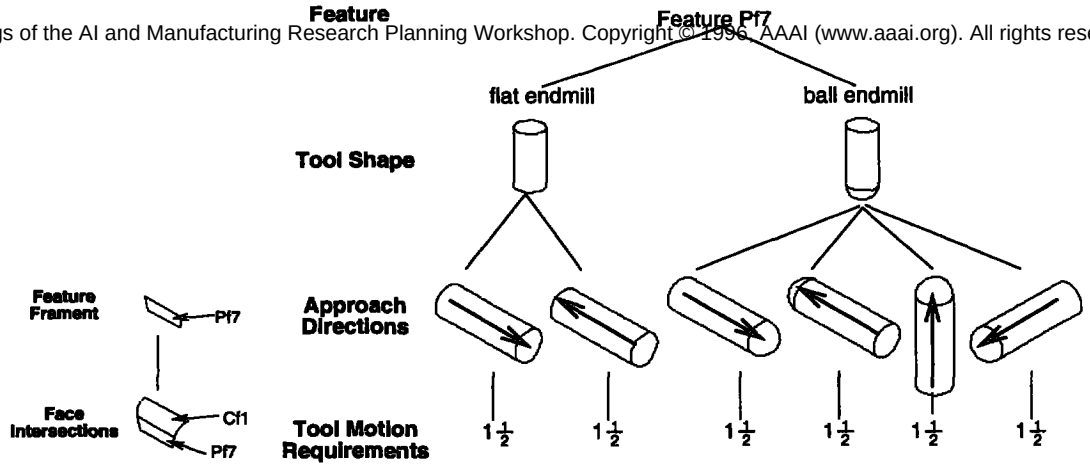
Figure 8: Feature and method development for Pf7 after tool shape knowledge.

of the tool axis and part surface information in the developing method description. If this is an initial feature fragment, as opposed to an aggregate feature, a removal volume is generated based on the part surface. Planar surfaces are swept in the direction of their normal until they meet with an obstacle or leave the delta volume. The resulting volume is swept in the direction of the tool axis. Non-planar surfaces, such as cylindrical and conical surfaces, result in cylinders and cones which are swept in the direction of the tool axis in a similar manner. Removal volumes for aggregate features are generated by combining their respective removal volumes. The global accessibility is checked at this point: if the removal volume intersects the part, then this feature is not globally accessible from this tool axis direction.

The volume generated for Pf7 is shown in Figure 10 (a). The methods for producing this feature have been further refined with the accessibility information. The method using a ball endmill from the back has been found to be invalid since the feature is not accessible from this direction.

Apply fixture knowledge In order to determine if a feature candidate is a valid one, it is also necessary to check if it is possible to hold the part while cutting the feature (or feature fragment). If the part can not be held, then the feature candidate must be eliminated. In the current implementation we consider only vises, but will consider other fixture types in future work. he goal of this stage is to determine if a valid fixturing method exists for the given feature candidate. However, full details of the fixture are not yet considered, only a few major aspects are examined: fixture type and clamping surfaces. The point is to determine only if a viable fixturing method exists, not to determine all details about it. Those details will be determined by the process planner. Using a variety of different levels of estimation is a powerful and important technique in planning and other types of problem solving, such as design. Many, if not most, non-fixturable features can be pruned early, on the basis of a very simple analysis.

Workpiece Representation. In order to perform fixture related reasoning on the geometric model it is necessary



(a) Pf7 feature (b) Pf7 methods

Figure 9: Feature and method development for Pf7 after tool motion knowledge.

to compute some additional fixture related concepts about the workpiece. Important relations from many types of fixtures include parallelism and coplanarity. These relationships between faces are computed and stored in a hierarchical structure. This allows us to prune large portions of the fixture search space by testing fixture requirements among groups of faces rather than particular instances of them. Figure 11 shows the workpiece representation for part I. (from Figure 2).

In this hierarchical structure, faces are first grouped into *parallel sets*; all faces belonging to this group have parallel normals. At the next level, faces are divided into *coplanar sets*. In each of these sets, all faces lie on the same plane. Finally, faces in a coplanar set are divided into two *coplanar subsets* according to the direction of their normal. For example, parallel sets are described by a vector parallel to the normals of all the faces in the set and coplanar sets may or may not be part of the boundary of the stock. Each node in the hierarchy has properties which are shared by all the faces under that node.

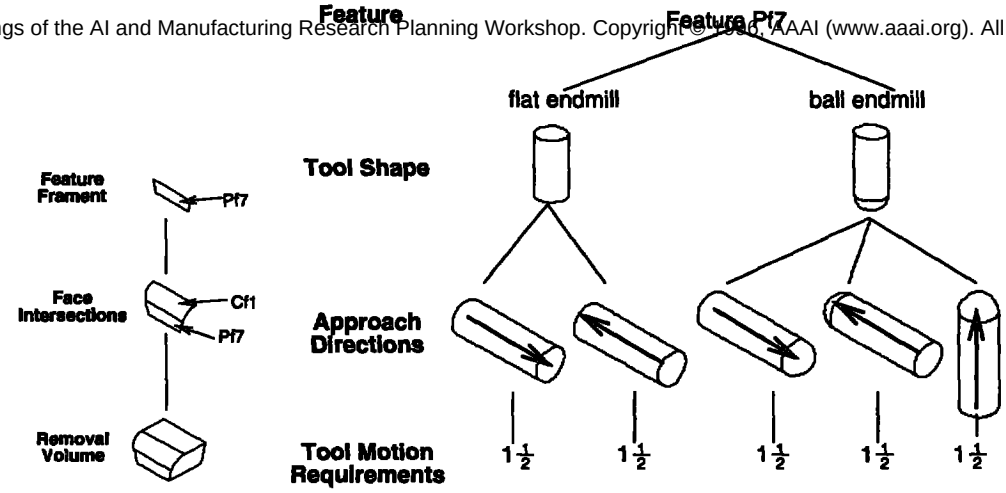
Inaccessible faces, on the inside of the part, which cannot be used as clamping surfaces, are not considered in the search process. Additionally, relations among the nodes are established. For example, two parallel sets may be perpendicular to each other, coplanar sets are separated by a particular distance, and coplanar subsets belonging to the same parallel set may have *NIH* (non-intersecting half spaces, a requirement for the clamping faces). These relationships are shown in Figure 11 as arcs between nodes. All this information is computed initially. More expensive information, such as the *overlap* between two faces (i.e., the intersection of two parallel faces projected on the same plane) is computed on an "as-needed" basis. Any new information gathered during analysis is attached to the nodes to save work in future analysis.

Fixture method generation. There are 2 subtasks in determining if an appropriate fixture exists for a feature fragment: Determining the seating face, and the clamping faces. A *seating face* is the face that the part will rest on. The *clamping faces* will be the faces gripped by the fixture. Each time a toolShapeMethod is generated, it is submitted to the *fixture method generator*. If no fixture method is found for it, it must be eliminated.

All the choices in the search process, from the vise type to the particular faces, are guided by heuristic knowledge. To show how the fixture method generator works, we will explain the results obtained for toolShapeMethod5, shown in Figure 7 (b).

Figure 11 shows the results of the search for a fixture by showing the portions of the workpiece representation activated during search. When the toolShapeMethod is submitted to the fixture method generator, the search process (for the vise) goes as follows:

1. Determining the seating face:
   (a) Sort all parallel sets with respect to the angle between their normal and the approach direction. Store them in SfPs (in the example SfPs = (Ps4 Ps2 Ps3 Ps1))
   (b) If SfPs is empty then no method exist for holding the part with a vise, otherwise select the first one (Ps4) and remove it from the list
   (c) Find all parallel sets perpendicular to the set selected in step 1b. If there are none, go to step 1b, else



(a) Pf7 feature (b) Pf7 methods

Figure 10: Feature and method development for Pf7 after volume generation.

sort with respect to the type of coplanar sets held by them (stck type is preferred) and store them in CfPs (for the example, CfPs = (Ps1))

(d) From the parallel set selected in step 1b find all the coplanar subsets whose normals make an angle of less than 90 degrees with the approach direction (so they can be used with a sine table). If there are none, go to step 1b, else sort by coplanar set type For the example, the result would be (Css5).

(e) All faces belonging to the coplanar subsets found in the previous step are candidate seating faces ((Pf2) in the case of the example)

2. Determining the clamping faces:

(a) If CfPs is empty go to step 1b, otherwise select the first parallel set (Ps1 for the example) and remove it from the list

(b) From the parallel set selected in the previous step, find all pairs of coplanar subsets with NIH (i.e.,non-interactive half spaces) relationship (in this case ((Css1,Css2))). If no pair was found, go to step 2a, otherwise sort them by distance and store them in CfCssPairs

(c) Each pairs of faces belonging to each pair in CfCssPairs is a pair of candidate clamping faces

(d) Propose the sets found in step 1e and 2c as candidates seating and clamping faces.

The result obtained for toolShapeMethod5 is that Pf1 and Pf6 are candidate clamping faces and Pf2 is a candidate seating face. Further analyses applied by the process planner (i.e, overlap, lateral accessibility and collision issues) will determine that these faces can be used for clamping the piece, as shown in Figure 12 (a). If all candidate solutions are rejected by these detailed analyses, the fixture method generator can expand the next set of candidate faces, informing that no fixture method exists if it runs out of alternatives to explore.

At this point, the set of methods is augmented by attaching fixture information. One feature fragment and approach direction may have several fixturing arrangements associated with it. Some methods may be pruned if no fixture configuration can be found for them. Like wise, the set of feature fragments can be pruned of all of its methods are pruned.

The final result of feature recognition and method generation for feature fragment Pf7 is shown in Figure 13. It turns out that most of the methods were pruned because there was no way to fixture the part in a vise for those approach directions. Figure 12 (b) shows an example of a method that was pruned because no good seating face could be found. The face against the bottom of the vise is angled and cannot seat the part properly. There was only one method for producing Pf7 with the available fixtures. The end result of applying feature identification and method generation to all the feature fragments is that each feature can be made with a method identical to the method found for Pf7 (Figure 13).

Group features

The purpose of this part of the process is to group features into larger features. Aggregate features are made by grouping valid features from the feature identification and method generation process. Criteria used

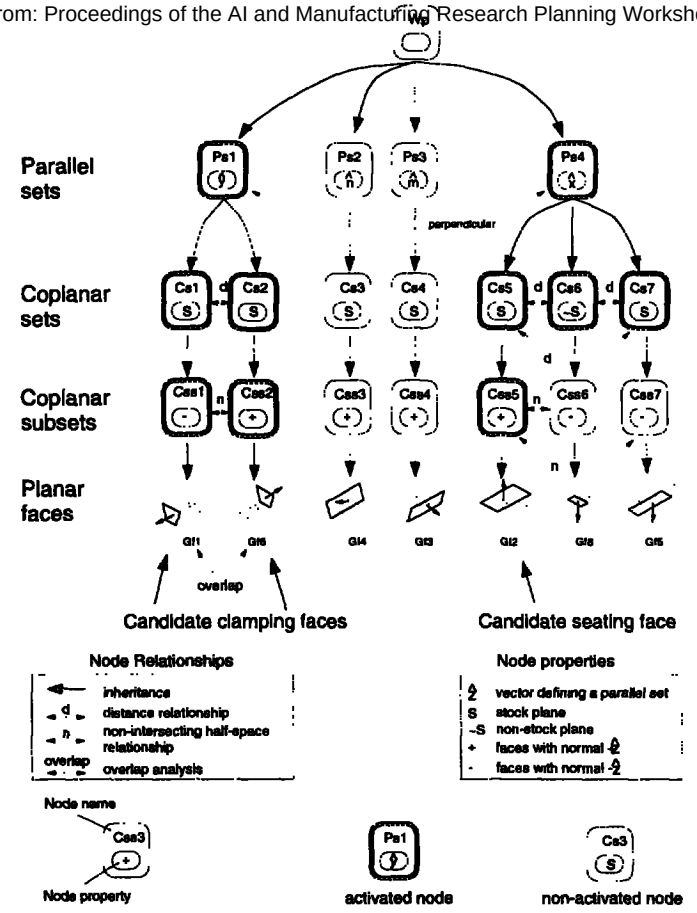
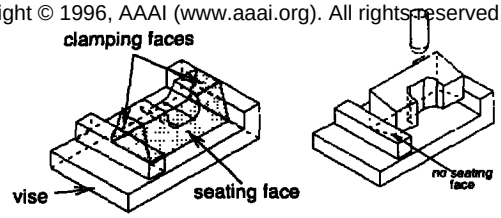


Figure 11: Workpiece representation.



(a) Valid. (b) Invalid.

Figure 12: Valid and invalid fixture methods.

to identify when two features should be combined include adjacency of surfaces and volumes, or common method parameters. For this example, after all grouping has been performed, the system finds that they can all be grouped into a single feature because all of the feature fragments form an adjacent set and they can all be made by a common method. Figure 14 shows the method which can be used to machine this feature.

Example 2: Part V

Figure 15 shows a more complex part adapted from (Vandenbrande & Requicha 1993). Figure 16 shows the features found by MEDIATOR along with the methods which can be used to produce them.

Conclusion

This paper has presented a method for producing a reconfigurable feature recognition system, which allows task-specific information to direct the recognition of features in a part model, while remaining maintainable and extendible. A description of machining equipment is used to simultaneously recognize features and generate manufacturing methods for producing those features. The system can be maintained and adapted easily by making changes to the equipment knowledgebase, which directly result in changes to the features output by the system.

The current implementation produces features and methods for 3-axis machined parts and can be customized to particular sets of tools available in a given shop. This is important because different shops must produce parts in different ways.

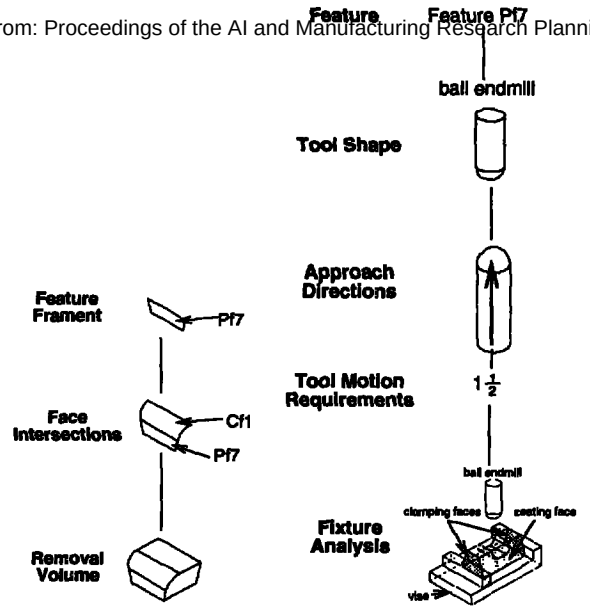
MEDIATOR in implemented in Scheme and C++ using the 3D Toolkit and ACIS from Spatial Technologies. In the future we hope to develop MEDIATOR to apply to a wider range of processes, such as 5-axis milling, turning and mill-turning. Additionally, we feel that MEDIATOR's methodology will be useful in developing an adaptable, maintainable CAD/CAPP integrations for other families of processes which are sensitive to the capabilities of the machining equipment, yet adaptable to changes in equipment technologies.

Acknowledgments

This work has been funded by NASA grant NAG-1-613 and NSF grant IRI92-090394, and is based on work developed under NSF I/UCR Center for Machine-Tool Systems Research.

References

Chang, T. C., and Anderson, D. C. 1992. Quick turnaround cell – an integration of feature based design and process planning. *1992 First Industrial Engineering Research Conference Proceedings.*



(a) Pf7 feature (b) Pf7 methods

Figure 13: Feature and method development for Pf7 after fixture knowledge.

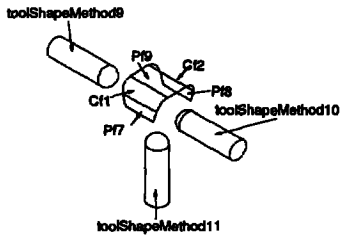


Figure 14: Tool shape method for highest level feature.

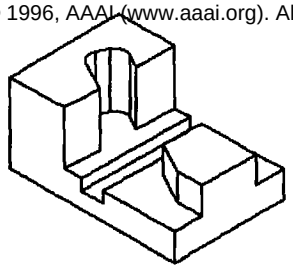


Figure 15: Part V.

Choi, B. K.; Barash, M. M.; and C., A. D. 1984. Automatic recognition of machined surfaces from a 3D model. *Computer-Aided Design* 16(2).

Cutkosky, M., and Tenenbaum, J. M. 1992. Towards a framework for concurrent design. *International Jorunal of Systems Automation: Research and Applications* 1(3):239–261.

Das, D.; Gupta, S. K.; and Nau, D. S. 1994. Estimation of setup time for machined parts accouting for work-holding constraints. *AII.*

Gaines, D. M.; Hayes, C. C.; and Kim, Y. S. 1995. Negative volume to process methods mapping for CAD/CAPP integration. In *Proceedings of IJCAI Intelligent Manufacturing Workshop*, 104–115.

Gupta, S. K.; Nau, D.; Regli, W.; and Zhang, G. 1994. Building MRSEV models for CAM applications. *Advances in Engineering Software* 20(2/3):121–139.

Joshi, S., and Chang, T. C. 1988. Graph-based heuristics for reconition of machined features from a 3D model. *Computer-Aided Design* 30(2).

Kim, Y. S. 1994. Volumetric feature recognition using convex decomposition. In Shah, J., et al., eds., *Advances in Feature Based Manufacturing.* Elsevier. chapter 3.

Marefat, M., and Kashyap, R. L. 1990. Geometric reasoning for recognition of three-dimensional object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(10):949–965.

Nau, D. S.; Gupta, S. K.; and Regli, W. C. 1995. AI planning versus manufacturing-operations planning: A case study. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2, 1670–1676.

Regli, W. C.; Gupta, S. K.; and Nau, D. S. 1994. Feature recognition for manufactruability analysis. In Shah, J. J.; Mantyla, M.; and Nau, D. S., eds., *ASME Computers in Engineering Conference*, 93–104.

Shah, J.; Shen, Y.; and Shirur, A. 1994. Determination of machining volumes from extensible sets of design features. In Shah, J., et al., eds., *Advances in Feature Based Manufacturing.* Elsevier. 129–157.

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

Woo, T. 1982. Feature extraction by volume decomposition. In *Conference on CAD/CAM Technology in Mechanical Engineering*, 76–94.

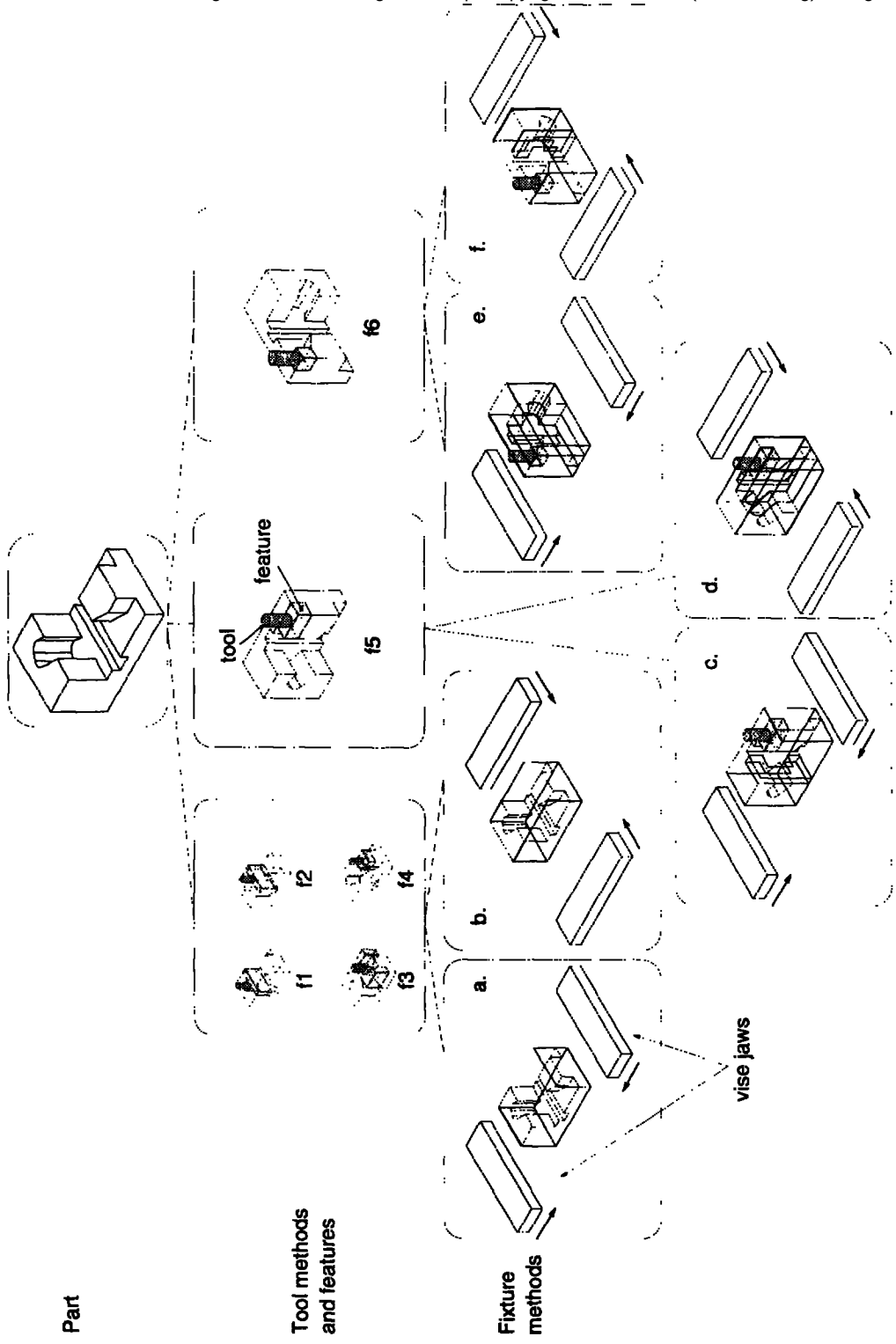


Figure 16: Results for part V (tool methods grouped by approach direction).