

# Intersecting features extraction from 2D orthographic projections

Rajan Ganesan† and Venkat Devarajan‡\*

This paper describes a new approach to solve the problem of intersecting features extraction. Feature extraction is important from several standpoints—automated process planning, NC machining, automated assembly and virtual prototyping. Feature extraction reported in the literature indicates limited success mostly in extracting a partial list of non-intersecting features. There has not been much success in the extraction of intersecting features. Techniques must continue to be developed to extract intersecting features and identify them as combinations of standard features. Our methods of intersecting feature extraction are different from current methods in that they operate on 2D CAD as input data, thus obviating analysis in complex 3D space. © 1998 Elsevier Science Ltd. All rights reserved

**Keywords:** feature-based CAD, feature extraction, intersecting features, 2D CAD, classification, divide and conquer

## INTRODUCTION

Feature-based CAD is an intelligent form of design representation in which the design is expressed in terms of some high-level definition that has direct relevance in various downstream activities. Features can be defined differently in many applications, e.g. automated process planning and NC machining, assembly, inspection and finite element analysis. Consequently, it is difficult to give a precise and single definition of a feature. The definition would be clear with a few examples—in the manufacturing domain, a feature representation is in terms of holes, slots, channels, fillets and chamfers. Traditional CAD represents the design in terms of either 2D entities (lines, arcs, circles), or 3D entities (wireframe, surfaces, solids), but these representations have little significance to the process planner who interprets the design in terms of manufacturing features, e.g. holes, pockets and fillets. Feature information allows the manufacturer to determine the machining tools

and manufacturing processes required to machine the part. This is the first reason to perform feature extraction.

Rapid changes in CAD technologies leave behind a huge quantity of legacy data that have to be exported into newer design environments. The problem is not one of data format conversion, but of different interpretations of the design and solid body representation methodology. Most new designs are modifications to older designs that exist as legacy data. Older designs could be available as 2D views, constructive solid geometry models, surface models or wireframe models, depending on the sophistication of the CAD system used. Feature extraction techniques have to be used to extract meaningful features from the non-feature-based CAD models. This is the second important reason to accomplish useful feature extraction.

## PROBLEM STATEMENT

Feature interactions are intersections of feature boundaries with those of other features such that either the shape or the semantics of a feature are altered from the standard or generic definition<sup>1</sup>. *Figure 1* shows a sample part that contains intersecting features. There are multiple interpretations of the intersecting features on this part, but one such interpretation is {(Pocket A  $\cap$  Square Slot), (Pocket B  $\cap$  Square Slot), (Step A  $\cap$  Square Slot), (Step B  $\cap$  Square Slot)} It is possible to have alternate interpretations for the same part.

Thus, our task is to recognize the presence of an intersecting feature, break it down to multiple individual features using at least one logical interpretation so that existing simple feature extraction systems can then be used to complete the feature extraction solution.

Most conventional methods used for the extraction of simple features fail with intersecting features. Feature intersections can cause faces to be shared between features without a clear demarcation and edges of faces may disappear or produce unexpected edges<sup>2</sup>. Thus, syntactic pattern recognition methods will fail to recognize intersecting features<sup>3</sup>. Other methods, e.g. Attribute Adjacency Graphs (AAG) fail to recognize an entire class of intersections<sup>2</sup>.

## LITERATURE SURVEY

Most of the research on feature-based CAD has concentrated on extraction of simple features. However, the

\*To whom correspondence should be addressed

†Automation and Robotics Research Institute, 7300 Jack Newell Blvd. S, Fort Worth, TX 76118, USA

‡Department of Electrical Engineering, P.O. Box 19016, The University of Texas at Arlington, Arlington, TX 76019, USA

Paper Received: 13 August 1997. Revised: 8 May 1998. Accepted: 27 May 1998

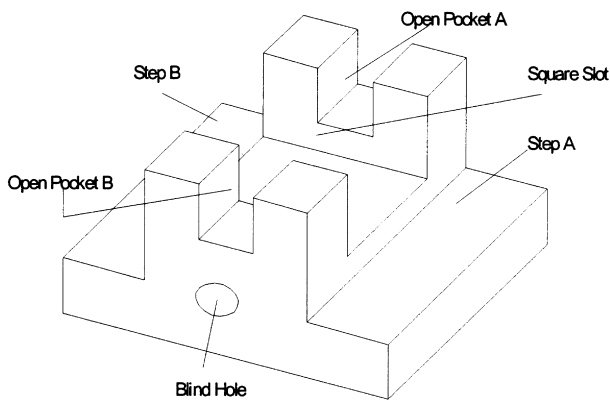


Figure 1 Example part with intersecting features

importance of intersecting features was not realized till a few years ago. The literature survey in this section will only discuss intersecting features. Several survey papers and books discuss the methods researched for non-intersecting features<sup>1-5</sup>.

Meeran and Pratt<sup>6</sup> proposed a preliminary approach to extract features from 2D orthographic views. The approach suggests recognition of lines and arcs that terminate in other feature patterns previously identified. This process has generally been defined as feature completion. The process is derived from the human process of recognizing intersecting features by extending surfaces and edges of objects. The method by Meeran and Pratt first generates hypothesis points inside the area of intersection. The area of intersection is split by feature completion. If the split areas constitute any standard feature, the process is complete. Otherwise, the algorithm selects a new pair of entities, finds a new intersection point and the process is iterated till standard features are recognized. The algorithms are limited to a fairly small set of simple intersecting features.

In the Attribute Adjacency Graph (AAG) method by Joshi and Chang<sup>7</sup>, BRep models are transformed into an AAG with faces as nodes and edges as arcs with assigned attribute values. Some simple interactions that can be handled include features that share a common edge and features that share a common face. Heuristics are used to extract many intersections.

Marefat and Kashyap<sup>8</sup> also use graph-based approaches on BRep models. In addition to face-edge relationships, they include the relative orientation of the faces. A rule-based system is used to examine the hypotheses generated by decomposing the graphs into maximal constituents. Only intersecting features that are recognizable by face unification can be extracted. Interacting features with disconnected face adjacency relationships cannot be recognized. The recognizer caters only to polyhedral objects and does not produce volume features. It does not appear to recognize pockets with arbitrary contours that include concave edges and islands<sup>3</sup>.

Kim<sup>9</sup> proposed a convex hull-based modified volume decomposition method. Although the system can recognize simple non-intersecting features, it can only handle intersections between features sharing a common edge. This limits it in the recognition of interacting features with a broken face adjacency relationship. In such cases, intersecting features may be recognized as a whole volume, not as individual features<sup>10</sup>.

Wang<sup>11</sup> used a backward growing approach by determining the volumes needed to grow the part into the

stock. All machined faces are considered candidates for possible growing bases and are prioritized according to some heuristics. A growing base is extruded to form a feature volume and this volume is unioned with the part to generate an intermediate part. This is done repeatedly till the part is transformed into the stock. The method works on fairly simple intersections between slots, blind slots, steps, blind steps, pockets, wedges, fillets and rounds. However, in many cases the intersecting features may be recognized as several small volumes that may not correspond to standard features. Also, the resultant set of feature volumes is not unique and dependent on the method of prioritizing the faces.

The method proposed by Sakurai<sup>12</sup> uses volume decomposition. The volume surrounding the object is decomposed into minimal convex cells. These cells are then combined and checked to see if the resulting volume is a standard feature. This method generates multiple interpretations of the intersecting features by allowing all possible combinations of the minimal cells. To reduce the combinatorial explosion problem, only combinations that produce the largest volume or the simplest shapes are considered valid. The method was then reformed to directly generate maximal convex cells (MCC). However, the number of interpretations increases dramatically as the number of MCCs increases. For example, eight MCCs can generate 7854 interpretations and 13 MCCs can generate billions. A second paper by Sakurai and Dave<sup>13</sup> extends the methods to solve intersections between curved surfaces. However, the decomposition still only generates all possible volumes and combining the maximal volumes to form features is still an open problem.

The method proposed by Tseng and Joshi<sup>10</sup> uses a combination of volume decomposition and reconstruction of feature volumes. During the volume decomposition process, the volume to be machined is identified and decomposed into small blocks (called basic removal blocks) by extending the boundary faces of the part. The input to the system is a BRep model and the stock is assumed to be the smallest rectangular prism that bounds the part. In the reconstruction phase, the basic removal blocks are combined systematically to produce feature blocks. Feature blocks may be generated by either 1D connections or 2D connection of basic removal blocks. The feature blocks are matched with the standard features to determine if they are standard features. There are multiple ways to connect the basic removal blocks depending on the direction of connection, and each can lead to a different interpretation. The method does not work when curved surfaces are present.

The recognizer by Vandenbrande and Requicha<sup>3</sup> can extract intersecting features of parts that can be machined on three-axis machining centers. This method also employs feature completion and surfaces are transformed to solids by the completion process. The input to the algorithms is a BRep model of the part and stock. The algorithm uses face geometry patterns to generate hints. Once a feature hint is found, the algorithm completes the feature by extending it to one or more directions without intruding the part. Portions of the completed feature are marked as optional or required to represent the feature interactions. The classification rules are logically complicated and are non-trivial for curved faces and non-orthogonal intersections<sup>3</sup>. The required portions of the feature represent the minimal volume that a cutting tool must sweep to generate the surface of the feature. The optional portion corresponds to the regions where a feature shares with other intersecting

features. In machining terms, the optional portion of the feature can be removed by machining either of the intersecting features. Features other than holes require additional processing in conjunction with feature completion. This is because a feature provided by a hint may not be sufficient to perform feature completion.

Karinthi and Nau<sup>14</sup> suggest algebraic solutions to the problem of multiple interpretations. The feature algebra developed by them can generate multiple sets of features from a given set of features. Features are not recognized by these algorithms. Feature operators, e.g. truncation, infinite extension and maximal extension are defined that operate on a pair of features recursively, generating a complete set of interpretations. The algebra is restricted to rectangular solids and cylinders that have their planar faces parallel to the faces of the stock.

The methods by Regli et al.<sup>15</sup> are based on the ‘hints’ methodology used by Vandenbrande termed as trace-based feature recognition. The work also mathematically formalizes the problem of feature extraction to a class of three-axis machining features. Techniques to reason completeness and computational complexity have been developed. Furthermore, parallel processing and distributed computation of the algorithms have been introduced for applications, e.g. collaborative virtual prototyping. The method can handle several real world parts that are available as BRep models. Han and Requicha<sup>16</sup> further developed the hints approach to integrate feature-based design and feature recognition.

### THE FLEXICAD ARCHITECTURE TO EXTRACT INTERSECTING FEATURES

FlexiCAD, developed by the authors, is a flexible feature-based CAD framework for transparently converting any CAD representation to feature-based CAD. The inputs to the system can be 2D CAD views, 3D constructive solid geometry models, 3D boundary representation models or 3D wireframe models. The core of the system is a unique feature extraction system in the 2D domain. Feature extraction in 2D is much simpler than in the 3D domain. This paper will show that this also holds true for extraction of complex intersecting features. FlexiCAD details can be found in other publications by the authors<sup>17-19</sup>.

Figure 2 shows the divide and conquer strategy used by FlexiCAD to handle intersecting features. This section also describes the classification of intersecting features used in this research. The object from which features are to be extracted is basically divided into three types of subparts. The root object is the 3D equivalent of the outermost loops in each view. Isolated subparts correspond to closed loops in each view that do not touch the outermost loop. Other closed

loops that have at least one edge touching the outermost loop correspond to non-isolated subparts. Firstly, all isolated subparts are identified and separated. These isolated subparts will either be protrusions or depressions. If the isolated subparts are holes or pockets, they are sent to the feature identification system for further analysis<sup>20</sup>. They are further classified as blind or through holes or pockets if they are depressions, or as rectangular or circular boss if they are protrusions. If the isolated subparts are neither holes nor pockets, they are classified as Class 1 intersection, i.e. intersecting features in the isolated subpart. Class 1 intersections will be handled differently and separated from the part. The resultant root object may contain either simple non-isolated features or intersecting features. Simple features on the root object include non-intersecting steps, open pockets, square slots, etc. These simple features are first extracted and the root object now contains only Class 2 intersecting features. Class 2 intersecting features are handled differently.

### DESCRIPTION OF THE METHODOLOGY

Figure 3 illustrates the methodology used by FlexiCAD to extract intersecting features. The system essentially consists of a preprocessor module and two other modules to extract type 1 and 2 intersections. Generation of the bounding box and outermost loops in each view is part of the preprocessing. Fillets are also isolated in the preprocessor to polygonize the 2D views of the object. Type 2 intersections require the extraction of cavities in each view.

#### Generating the bounding box

All three orthographic views are enclosed by a rectangular bounding box. The bounding box is generated by starting at a reference point at the lower left corner of the view and tracing a rectangle by locating the maximum values of the *x* and *y* coordinates of the entities in that view. If the outline of the object is circular, then critical points corresponding to the extrema of the circle are used to generate the bounding box.

#### Feature library

The feature database contains the following non-isolated features: square slot, dovetail slot, V-slot, T-slot, open pocket, step, notch, wedge, fillet and partial cylinder. Figure 4. shows a complete list of isolated and non-isolated features in the database. By inspecting several realistic mechanical parts, it is found that this set of features is a reasonably complete set and any other composite feature can be represented in terms of this set of features.

#### Constraint propagation

Constraints are developed for all features in the feature database to unambiguously describe the feature. In 2D, the correspondence between the three views can be used to develop these constraints. The constraints for a simple V-slot in the front view are given below. However, the feature can exist in any of the three views with any orientation to the axes. Detailed constraints for the other features are available in other publications by the authors<sup>18</sup>.

The top, front and side orthographic views corresponding to a V- slot on the root object are shown in Figure 5. In the

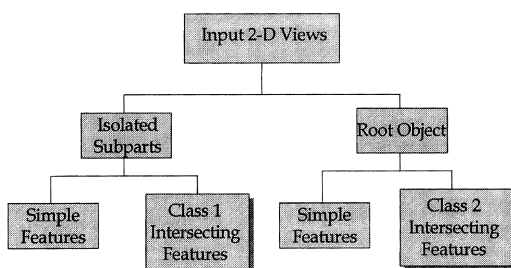


Figure 2 The divide and conquer approach

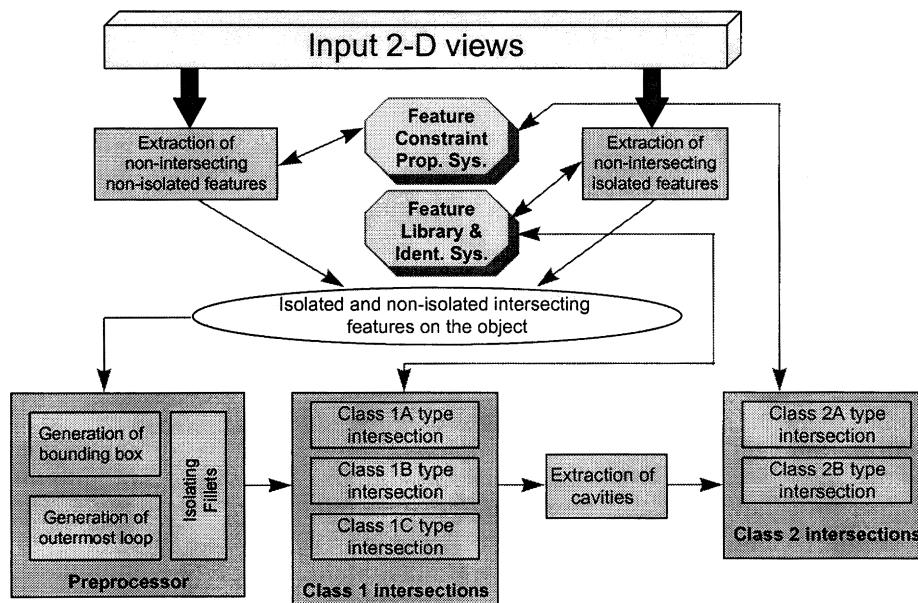


Figure 3 Overall approach to the extraction of intersecting features

example under consideration, the cavity is found in the front view. The constraints are as follows:

- (1) The cavity is three sided and starts and ends on the same edge of the bounding box.
- (2) The three sides of the cavity are not perpendicular to each other and the angles made at corner points B3 and C3 are greater than 90°.
- (3) A line A1B1 is found in the top view with its x

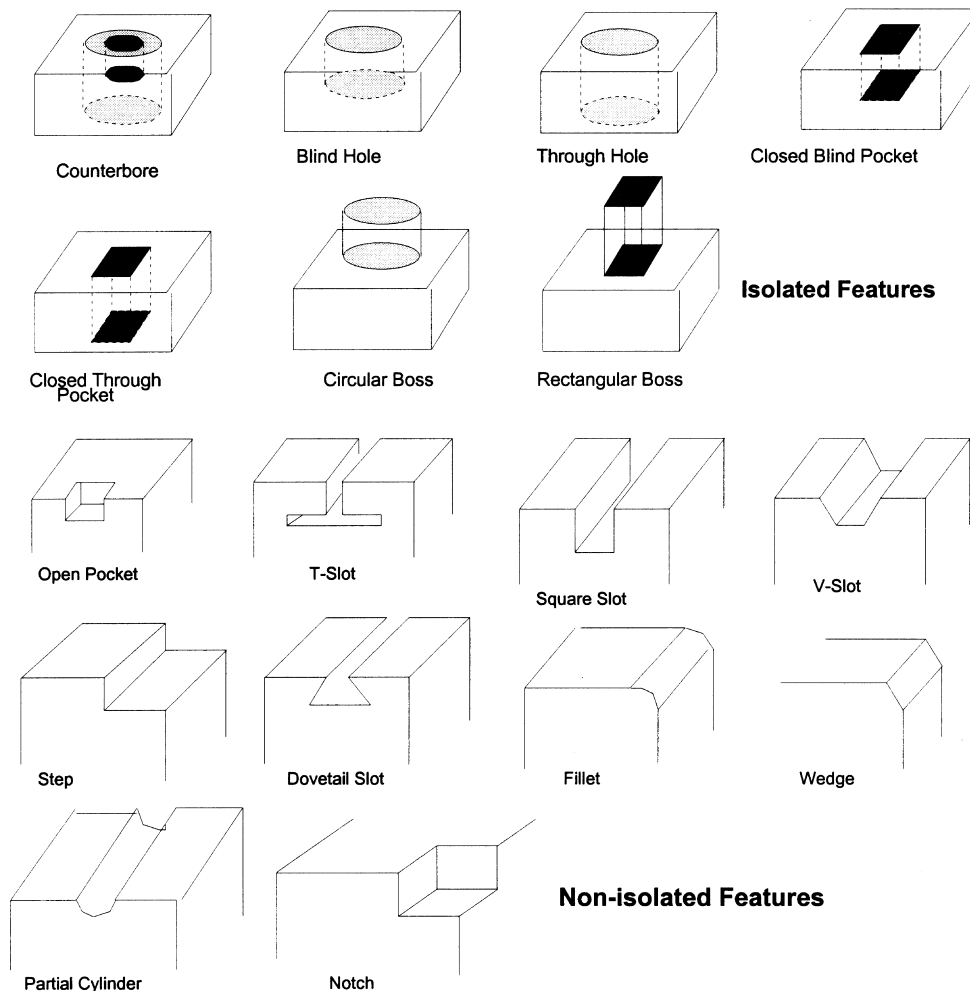


Figure 4 Features in the database

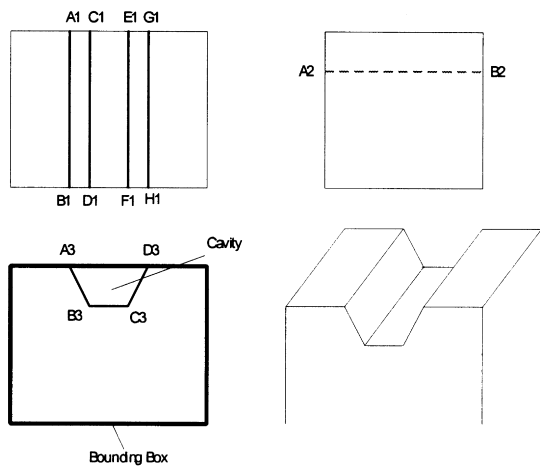


Figure 5 Constraint propagation for a V-slot

coordinates matching the  $x$  coordinates of point A3 in the front view. A line C1D1 is found in the top view with its  $x$  coordinates matching the  $x$  coordinates of point B3 in the front view. A line E1F1 is found in the top view with its  $x$  coordinates matching the  $x$  coordinates of point C3 in the front view. A line G1H1 is found in the top view with its  $x$  coordinates matching the  $x$  coordinates of point D3 in the front view.

- (4) Lines A1B1, C1D1, E1F1 and G1H1 are parallel in the top view.
- (5) Lines A1B1, C1D1, E1F1 and G1H1 start and end on two parallel edges of the bounding box.
- (6) A dashed line A2D2 is found in the side view with its  $z$  coordinates matching the  $z$  coordinates of line B3C3 in the front view.
- (7) The  $y$  coordinates of point A2 in the side view match the  $y$  coordinates of points B1, D1, F1 and H1 in the top view.
- (8) The  $y$  coordinates of point B2 in the side view match the  $y$  coordinates of points A1, C1, E1 and G1 in the top view.

### Isolating fillets

Fillets require complex plane intersection calculations because of their curved surfaces. However, working in 2D, it is comparatively easy to isolate the fillets initially before any intersecting features are extracted. Fillets present arcs in any one of the three views. In general, fillet arcs in 2D have included angles less than or equal to  $90^\circ$ . Other curved features with greater included angles are probably intersecting holes. If the arcs satisfy this test, they are isolated and placed in the list of fillets with the appropriate fillet radius and location. The arc entity in the view is now deleted and replaced with a pair of virtual straight line edges between the end points of the arc. After all the arcs are isolated, the root object is basically planar and easier to handle.

### Generating the outermost loop

Before the outermost loop in each view is located, all polyline entities in the input views are broken into simple line entities. This means that if there are two lines in a view that intersect, they are broken down into four simple lines. The outermost loop is generated by starting at a reference

point at the lower left corner (origin) of the view and tracing all the line entities connected to the first reference line. The line at maximum angle with the reference line is selected into the outermost loop. This is continued till the reference point is reached again.

### Locating cavities

Cavities are the closed loops corresponding to areas that are generated by subtracting the area of the outermost loop from the area of the bounding box. A cavity can be detected by tracing all line segments on the outermost loop and comparing the edges with the bounding box. When an edge breaks away from the bounding box, a cavity is established. The cavity is closed when an edge of the outermost loop reconnects to the bounding box. Cavities are the starting blocks in the extraction of Class 2 intersecting features from the root object.

### Analysis on Class 1 intersections

Class 1 intersections can be further classified based on whether the isolated subpart is a depression or protrusion:

- (1) Class 1A: holes intersecting holes/circular boss intersecting circular boss
- (2) Class 1B: holes intersecting closed pockets/circular boss intersecting rectangular boss
- (3) Class 1C: closed pockets intersecting closed pockets/rectangular boss intersecting rectangular boss.

It must be noted here that the above three possibilities can lead to several other combinations because holes and pockets can both be blind or through.

From the 2D point of view, the extraction of intersecting features from isolated depression subparts is the same as extraction from isolated protrusion subparts. Other methods are available to detect if the subparts are protrusions or depressions, and if they are blind or through<sup>20</sup>. The feature intersection analysis here will only concentrate on splitting the loops associated with the isolated subparts into generic features.

### Class 1A intersections

Figure 6 shows an example of Class 1A intersection. If any arcs with included angle greater than  $90^\circ$  are found as part of the isolated subpart, then that points to the existence of a hole or circular boss. In 2D, this is very easy to implement because all neutral file formats represent arcs uniquely.

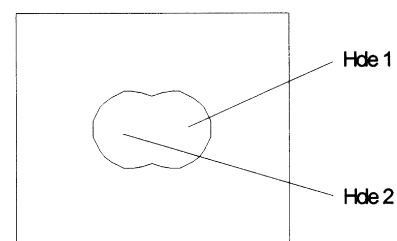


Figure 6 An example of Class 1A intersection

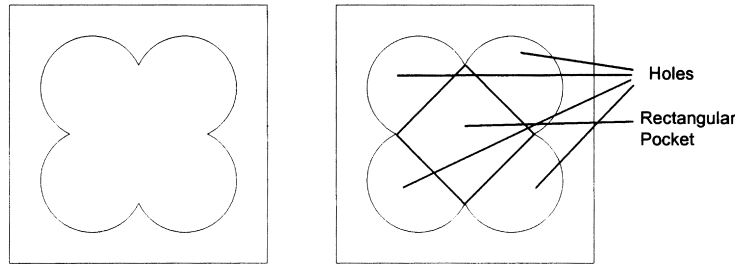


Figure 7 An example of Class 1B intersection

**Class 1B intersections**

An example of Class 1B type intersection is shown in Figure 7. The analysis required to break this subpart is as follows:

- (1) Search all isolated closed loops to find loops that contain both arcs and lines.
- (2) Arcs in a loop constitute intersecting holes or circular boss. Extract and store separately.
- (3) Modify the loop by replacing the arcs with straight line edges between the arc ends (called virtual edges).
- (4) Check the resulting loop to see if it is a standard feature (pocket or rectangular boss). If it is not, then it is now a Class 1C type intersection. Continue to process the loop. If the resultant loop is a standard feature, extract and save it separately. Delete the isolated loop from the drawing.

For the example in Figure 7, four holes and one pocket are found (assuming they are depressions). They are also not yet classified as through or blind. The pocket was readily extracted as a pocket. However, in Figure 8 that shows a common keyway, the pocket is not readily extracted after the hole is isolated. This now falls under Class 1C type intersection.

**Class 1C intersections**

In the simplest situation, the loops corresponding to the features are intact. In this case, each loop (if simple) will correspond to one feature. An example for this type of intersection is shown in Figure 9. This represents two intersecting pockets. The algorithm for these simple intersections is as follows:

- (1) Form all possible combinations of four-sided loops with the perpendicularity test.
- (2) Examine each loop to see if it corresponds to a valid feature.

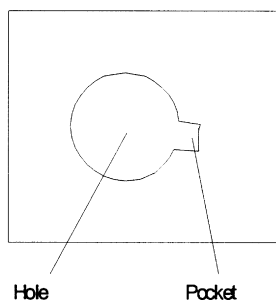


Figure 8 An example of complex Class 1B intersection

- (3) If it does, then break each loop and save separately as a pocket/rectangular boss. If it does not, try a different combination of loops. Check again.

**Perpendicularity test**

For two edges A and B that are parallel, if the parallel projections of the end points of A on edge B, lie on edge B, then the edge A satisfies the perpendicularity test.

An example to illustrate the algorithm is shown in Figure 10. The subpart corresponds to two intersecting pockets (assuming depressions). Although the subpart can be broken down to five pockets, the method of combination will solve this intersection as just two pockets in the best case and as four pockets as alternate interpretation. The loop to be analyzed contains 12 edges (1–12).

Figure 11 illustrates the different pockets that can be extracted from the compound pocket using the perpendicularity test.

**Analysis on Class 2 intersections**

Similar to Class 1 intersecting features, Class 2 intersecting features can also be classified into two subclasses based on intersecting 2D entities in the cavities extracted. They are as follows:

- (1) Class 2A: arcs intersecting lines in the cavity.
- (2) Class 2B: lines intersecting lines in the cavity.

Class 2A intersections are handled exactly like Class 1B intersections and the arcs are separated from the loops to form polygonal loops. These will then be handled as Class 2B intersections.

A hierarchical analysis is used in solving Class 2 intersections. The hierarchy is based on the concept of open and closed cavities and their related features. The following observations can be made:

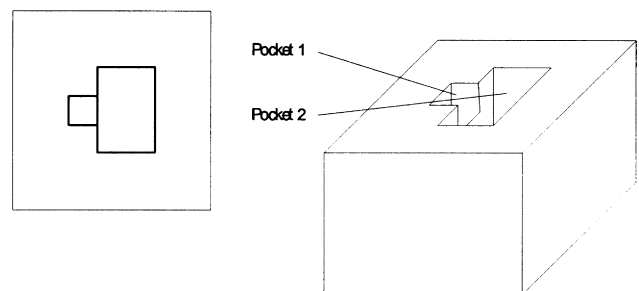


Figure 9 An example of Class 1C intersection with depressions

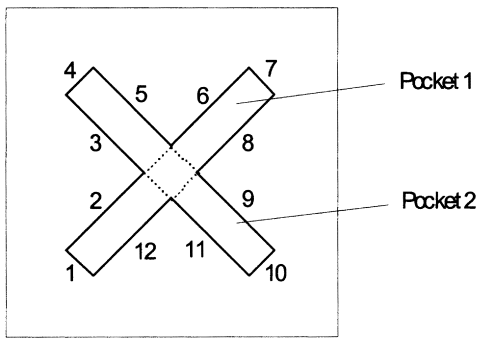


Figure 10 Intersecting pockets in the subpart

- (1) Compound open cavities correspond to intersections among non-isolated features that are also open-slots (T, V, dovetail and square), step and wedge/chamfer.
- (2) Compound closed cavities correspond to intersections among non-isolated features that are also closed—open pocket and notch.
- (3) There are also two forms of compound open cavities—cavities that start and end on the same face of the outermost loop, and cavities that start and end on perpendicular faces of the outermost loop. This further narrows down the features to all forms of slot intersections in the first category, and steps and chamfers or wedges in the second category.

- (4) In the category that contains wedges, chamfers and steps, a distinction can again be made based on the number of sides and angle made by the cavity. Both chamfers and wedges are inclined open cavities, whereas a step forms an orthogonal open cavity.
- (5) In the slot category, the different types of slots can be distinguished by using constraint propagation, and the number of cavity edges and their relative angles are important considerations.
- (6) There are also two forms of compound closed cavities—cavities that start and end on the same face of the outermost loop, and cavities that start and end on perpendicular faces of the outermost loop. This narrows down the features to open pocket intersections in the first category and notches in the second category.

### Class 2B intersections

Figure 12 illustrates examples of the four types of Class 2B intersections. Each type of intersection has a dominant feature that intersects others in the cavity. The dominant feature is a square slot in the first type, step in the second type, open pocket in the third and notch in the fourth type. In each case, the perpendicularity test is used to split the cavity into constituent features.

The algorithm to extract Class 2 intersections is as follows:

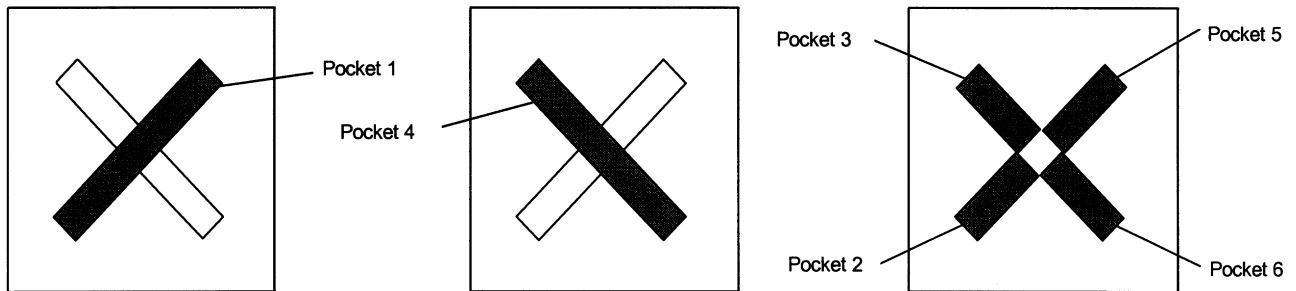


Figure 11 Alternate sets of pockets extracted from the part.

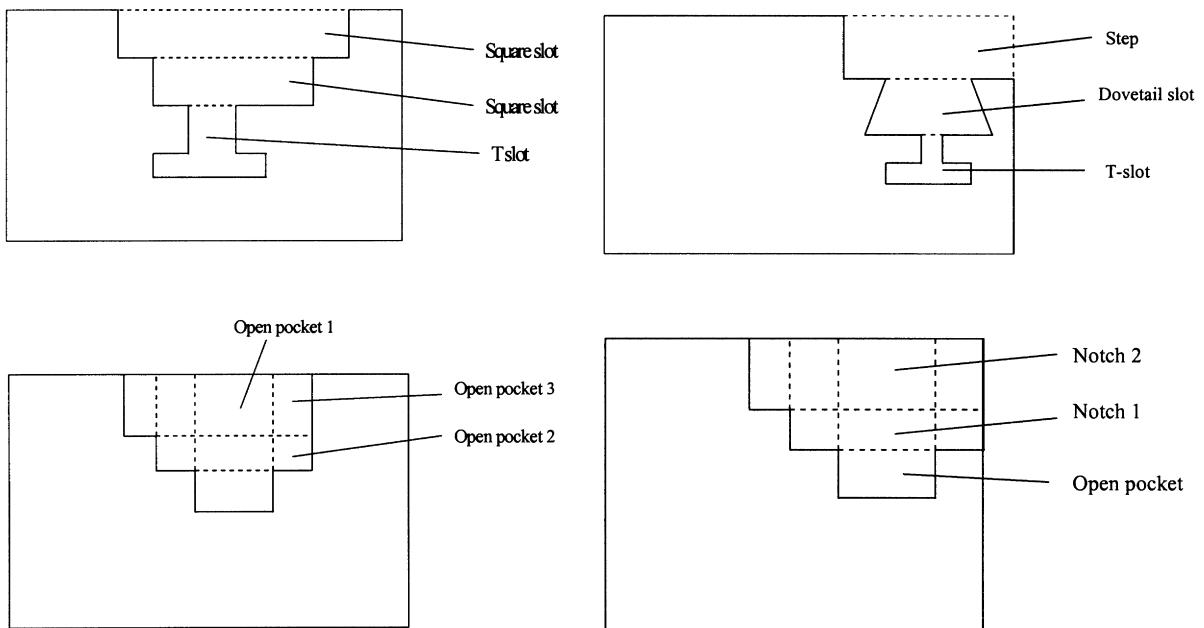


Figure 12 Types of open and closed cavities

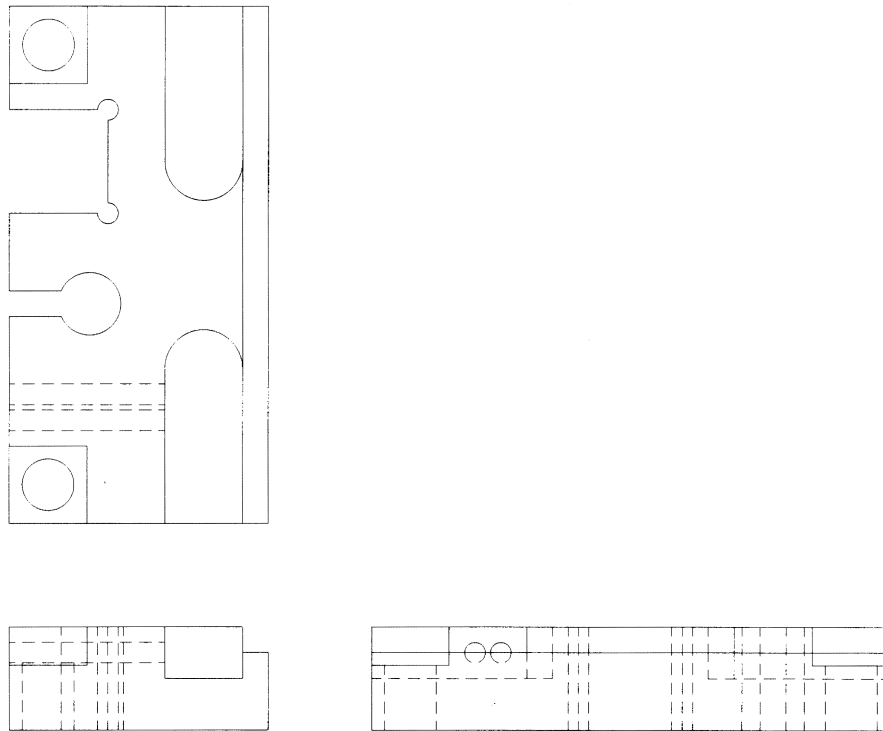


Figure 13 Original orthographic views of the part

- (1) First search for T-slots in the cavity using constraint propagation. Extract the T-slots first because they will be broken down into other features later.
- (2) Remove the edges corresponding to the T-slot and replace them with one edge between the two ends.
- (3) Process the cavity and apply the perpendicularity test.
- (4) Each time the test is satisfied, constraint propagate the extracted loop to identify a feature.
- (5) Once the feature is identified, remove the constituent edges and replace them with a single edge between the ends.

- (6) Re-analyze the cavity till the complex cavity is completely closed.

### ILLUSTRATIVE EXAMPLE

This section illustrates with figures some of the above algorithms on a part that contains a few intersecting features. *Figure 13* is the original input orthographic views. *Figure 14* illustrates the extraction of through

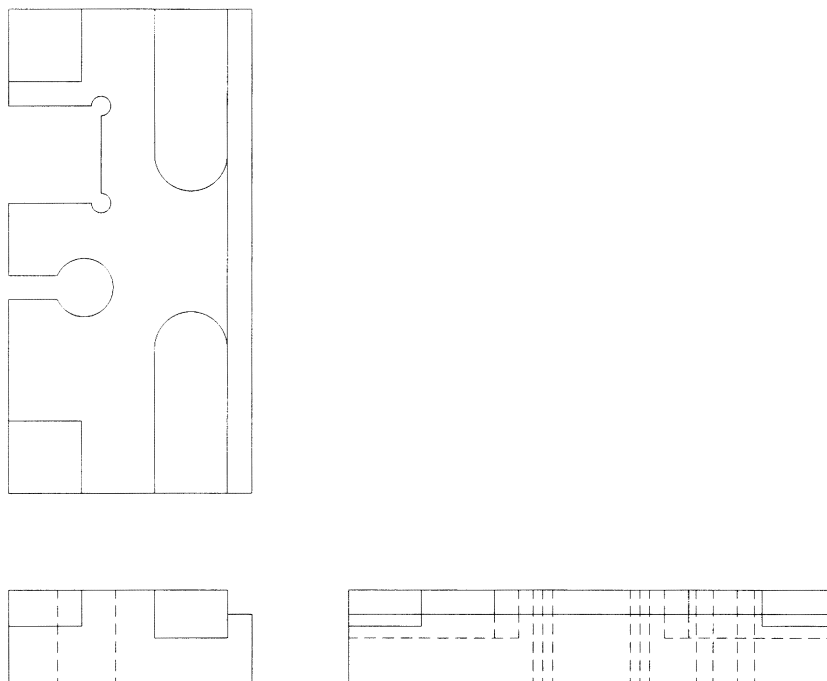
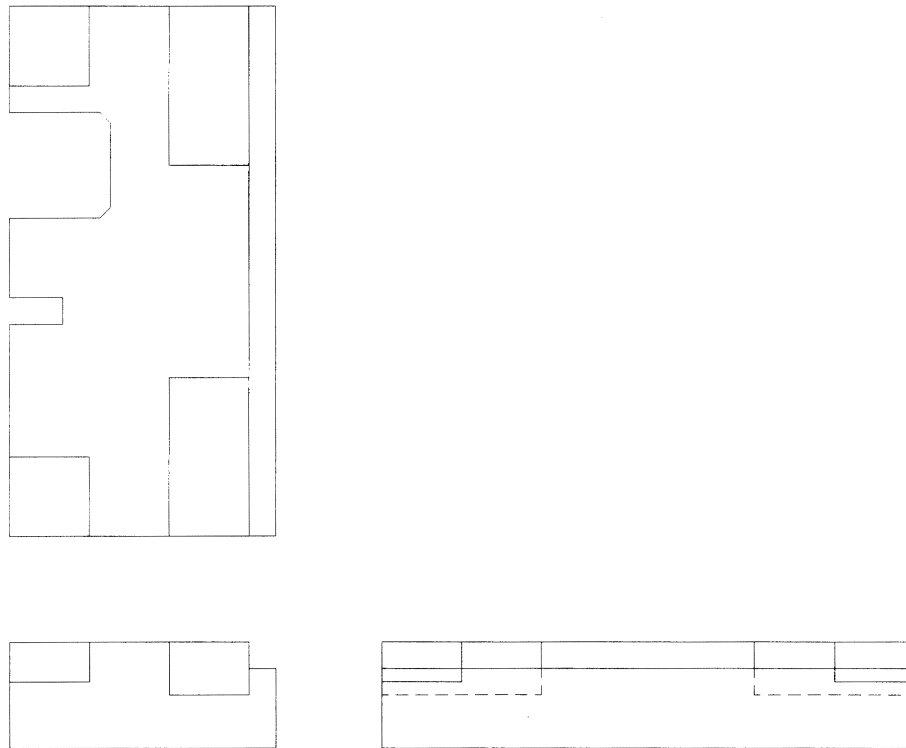


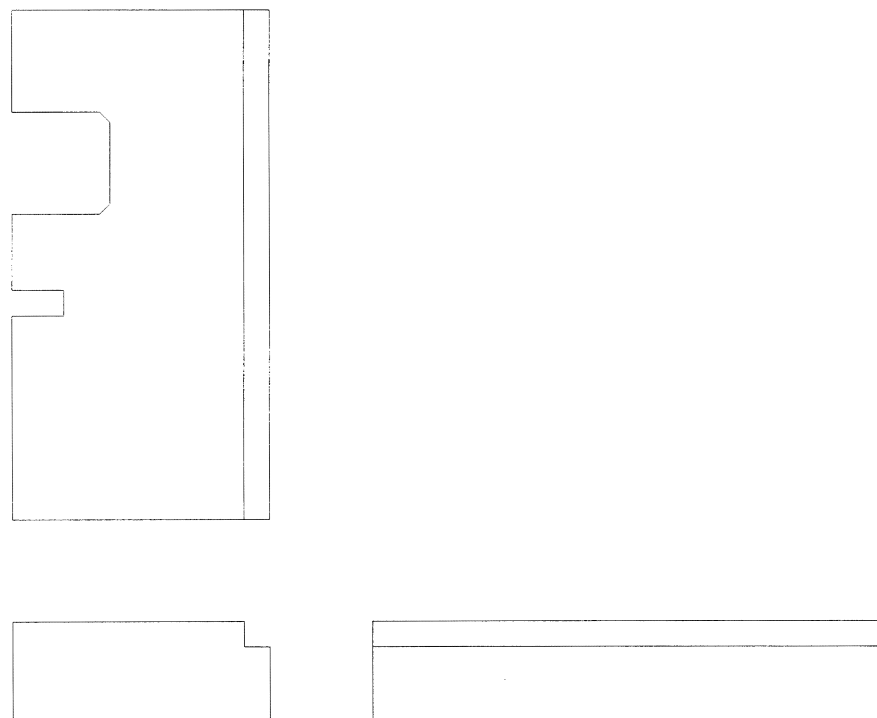
Figure 14 Extraction of holes by isolated subpart extraction



**Figure 15** Extraction of intersecting holes

holes. The two circles found in the top view and side view correspond to isolated subparts. They are identified as holes and then isolated from the drawing. In *Figure 15*, the fillets and rounds are extracted and the arcs representing them in the views are replaced with virtual edges between the ends. In *Figure 16*, the cavities corresponding to the open pockets are found in all three views. For each pocket in a view, the depth of the open pocket can be found from the other two

views. Finally, in *Figure 17*, the bounding box is drawn around each view. In the top view, two cavities A and B are found. These cavities are matched with the constraints to identify them as square slots. Similarly, cavity C found in the side view is identified as a step. *Figure 18* shows the final feature-based 3D output of the system. The features identified are—seven through holes (1, 4, 5, 6, 9, 15, 16), two open pockets (8, 13), two notches (7, 14), two square



**Figure 16** Extraction of the notches

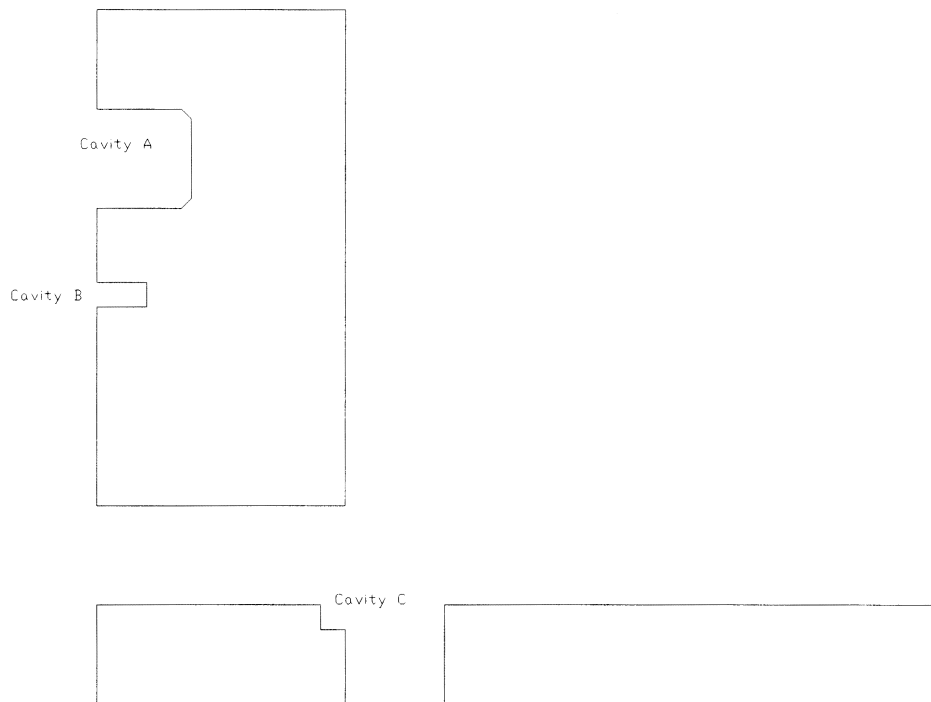


Figure 17 Extraction of the step and square slots

slots (2, 3), one step (12), and two fillets (10, 11). Geometric attributes for each feature are calculated and stored in the database of extracted features.

The intersections on the object can be represented as follows:  $\{(1 \cap 14), (6 \cap 7), (9 \cap 3), (16 \cap 3), (2 \cap 15), (4 \cap 8), (5 \cap 8), (11 \cap 13 \cap 12), (8 \cap 10 \cap 12)\}$

### ADVANTAGES AND LIMITATIONS OF THE APPROACH

The algorithms discussed in this paper have several advantages over other methods suggested in the literature. The advantages mainly arise from two issues—firstly, the approach to working in the 2D domain; and secondly, the divide and conquer approach. The following are some notable merits:

- (1) Unlike other methods in the literature, these algorithms operate in 2D space. This has dual significance. Firstly, the algorithms have turned out to be simpler and easier from the implementation point of view; and secondly, a

- vast number of existing drawings in 2D can be converted completely to feature-based CAD, which cannot be accomplished by any of the existing systems.
- (2) Although most extraction systems today only use BRep as input data, there is still a large volume of other 3D forms, e.g. CSG, or even simple wireframe. These algorithms work with any of the 3D representation schemes as input.
- (3) The divide and conquer strategy has also not been used by other researchers, who do not extract features in any preferable order. With this approach, simple features are extracted first, leaving the most complicated intersections to the end.
- (4) The method can handle intersections between curved surfaces and also extract simple curved features, e.g. fillets and rounds.
- (5) The method does not use any special functionality available in specific CAD systems. This makes the system self standing and can be integrated with any commercial CAD software.

The methods suffer from some potential problems. Some

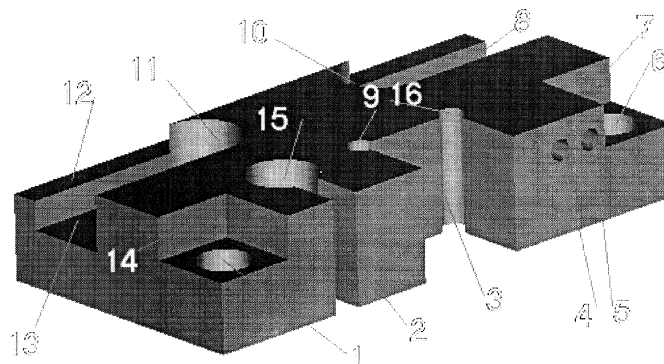


Figure 18 Final 3D feature-based part

known limitations are:

- (1) Currently, the algorithms are not capable of handling complex non-orthogonal intersection for some features. These must be handled by extending the constraint propagation techniques used here to cross-sectional and auxiliary views.
- (2) Features cannot be extracted from freeform surface objects due to the 2D approach.

## CONCLUSIONS

This paper describes a novel approach to extracting intersecting features from 3D models through intermediate orthographic views. The method uses a divide and conquer approach that reduces the complexity of the problem at various stages. The methods have been proved to handle a wide range of objects that can be represented uniquely by three orthographic views. The algorithms have been verified using a number of sample parts from the NIST Process Planning Repository. Detailed results of feature extraction from these parts are available in Ref.<sup>18</sup>.

## ACKNOWLEDGEMENTS

The authors would like to thank Raytheon/E-Systems, Aerospace Agile Manufacturing Research Center, and Automation and Robotics Research Institute of the University of Texas at Arlington for financial support on this research.

## REFERENCES

1. Shah, J.J. and Mantyla, M., *Parametric and Feature-based CAD/CAM-Concepts, Techniques and Applications*, John Wiley, 1995.
2. Allada, V. and Anand, S., Feature-based modelling approaches for integrated manufacturing: state-of-the-art survey and future research directions. *International Journal of Computer Integrated Manufacturing*, 1995, **8**, 411–440.
3. Vandenbrande, J.H. and Requicha, A.A.G., Geometric computation for the recognition of spatially interacting machining features. In *Advances in Feature Based Manufacturing*, Elsevier Science B.V., 1994, pp. 83–106.
4. Case, K. and Gao, J., Feature technology: an overview. *International Journal of Computer Integrated Manufacturing*, 1993, **6**, 2–12.
5. Solomons, O.W., Houten, F.J.A.M. and Kals, H.J.J., Review of research in feature-based design. *Journal of Manufacturing Systems*, 1992, **12**, 113–132.
6. Meeran, S. and Pratt, M.J., Automated feature recognition from 2D drawings. *Computer Aided Design*, 1993, **25**, 7–17.

7. Joshi, S. and Chang, T.C., Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer Aided Design*, 1988, **20**, 58–66.
8. Marefat, M. and Kashyap, R.L., Geometric reasoning for recognition of three-dimensional object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990, **12**, 949–965.
9. Kim, Y.S., Convex decomposition and solid geometric modeling. Ph.D. Dissertation, Stanford University, 1990.
10. Tseng, Y.J. and Joshi, S.B., Recognizing multiple interpretations of interacting machining features. *Computer Aided Design*, 1994, **26**, 667–688.
11. Wang, M.T., A geometric reasoning methodology for manufacturing feature extraction from a 3D CAD model. Ph.D. Dissertation, Purdue University, 1990.
12. Sakurai, H., Volume decomposition and feature recognition: Part 1—polyhedral objects. *Computer Aided Design*, 1995, **27**, 833–843.
13. Sakurai, H. and Dave, P., Volume decomposition and feature recognition: Part 2—curved objects. *Computer Aided Design*, 1996, **28**, 519–537.
14. Karinithi, R.R. and Nau, D., An algebraic approach to feature interactions. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1992, **14**, 469–484.
15. Regli, W.C., Gupta, S.K. and Nau, D.S., Extracting alternative machining features: an algorithmic approach. *Research in Engineering Design*, 1995, **7**, 172–192.
16. Han, J. and Requicha, A.A.G., Integration of feature based design and feature recognition. *Computer Aided Design*, 1995, **29**, 393–403.
17. Ganesan, R. and Devarajan, V., A feature-based framework for transforming and representing multiple format CAD for virtual prototyping. In *Virtual Prototyping: Virtual environments and the product design process*, Chapman and Hall, 1995, pp. 129–145.
18. Ganesan, R., Automated recognition of intersecting features from 2D CAD for collaborative virtual prototyping. Ph.D. Dissertation, The University of Texas at Arlington, 1997.
19. Ganesan, R. and Devarajan, V., An approach to extracting intersecting features from 2D CAD. In *1997 ASME International Mechanical Engineering Congress and Exposition, Concurrent Product Design and Environmentally Conscious Manufacturing*, 1997, pp. 79–94.
20. Tyan, L.W. and Devarajan, V., Automatic identification of non-intersecting machining features from 2D CAD input. *Computer Aided Design*, 1998, **30**(5), 357–366.

Rajan Ganesan obtained his Masters and Ph.D. degrees in Electrical Engineering from the University of Texas at Arlington and has been associated with the Automation and Robotics Research Institute since 1992. He is currently employed as Director for CAD Products with Imagecom Incorporated. His research interests are in automated CAD representation conversion, legacy data insertion into collaborative virtual prototyping environments and feature-based manufacturing.

Venkat Devarajan is Professor in Electrical Engineering at the University of Texas at Arlington. He worked in the aerospace industry for over a decade. His experience is in image processing applications to aerospace and automated manufacturing. He helped develop photo-based visual systems technology, which has since found wide acceptance in the flight simulation community. His present research interests include visual systems, image processing and photogrammetry applications to aerospace and manufacturing.