A Methodology for automatic retrieval of similarly shaped machinable components – Research Update

Mark Ascher - Dept of ECE

Chandan Pitta – Dept of ECE

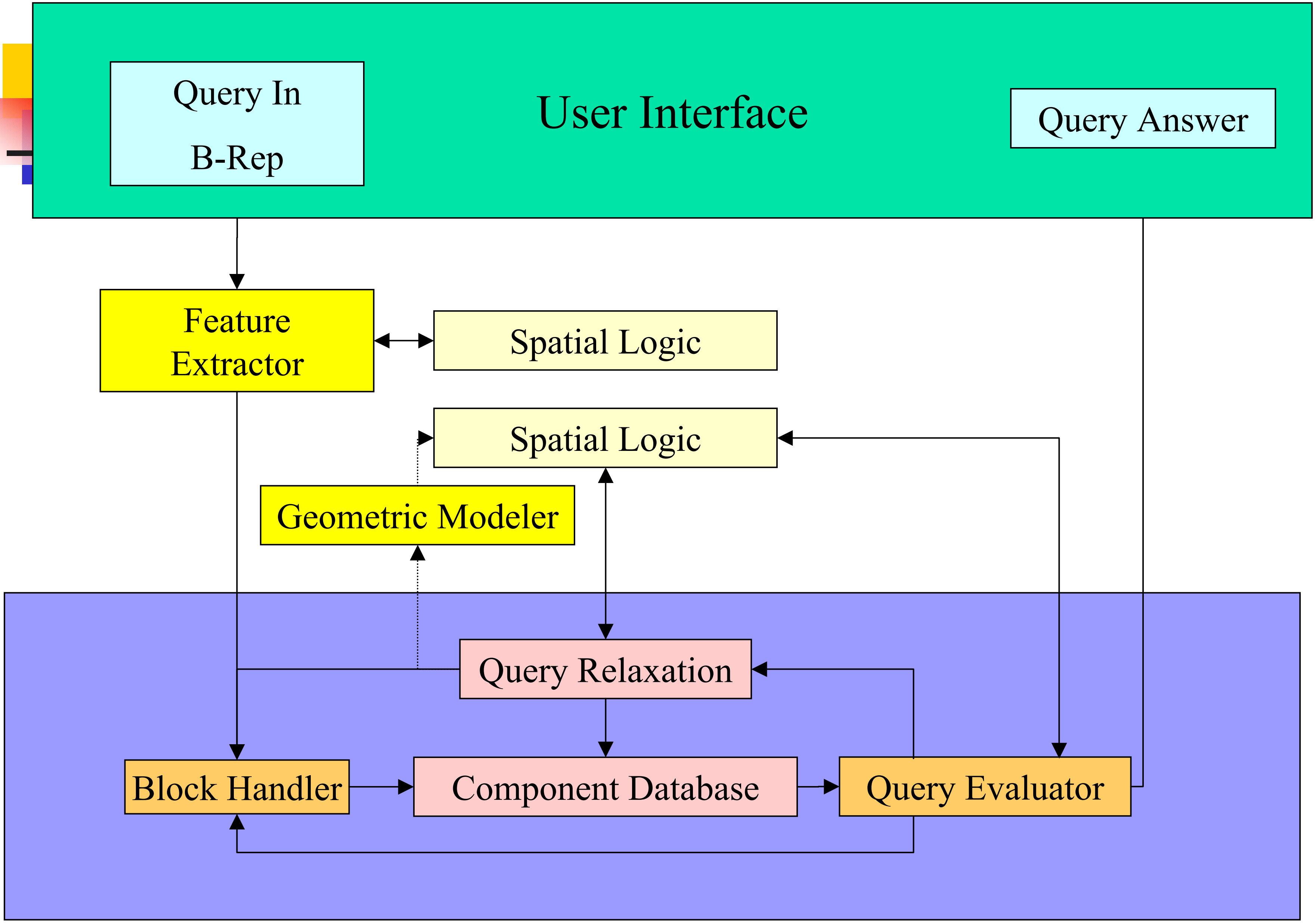
Motivation

- Retrieval of similarly shaped components can:
  - Add functionality to existing CAD databases
  - allow for the reuse of process plans which can both speed up and reduce the cost of development.

Challenges

- Retrieval of similarly shaped components has many challenges:
  - Multiple interpretations
  - Interacting features
  - Topological differences do not guarantee component dissimilarity
  - Graph matching solution is computationally intensive

System Overview



Related Work

- Shape Based Similarity Retrieval (Eakins)
  - Two dimensional parts
  - retrieved complete components
- Volumetric Reasoning (Lee et al) and Planar Reasoning (Cohn et al)
  - Groundwork for symbolic volumetric reasoning
  - Does not address part matching
- Content Retrieval From Images Based on Knowledge of Shape (Hsu et al)
  - Worked with medical images
  - Presented the Type Abstraction Hierarchy Concept

Related Work

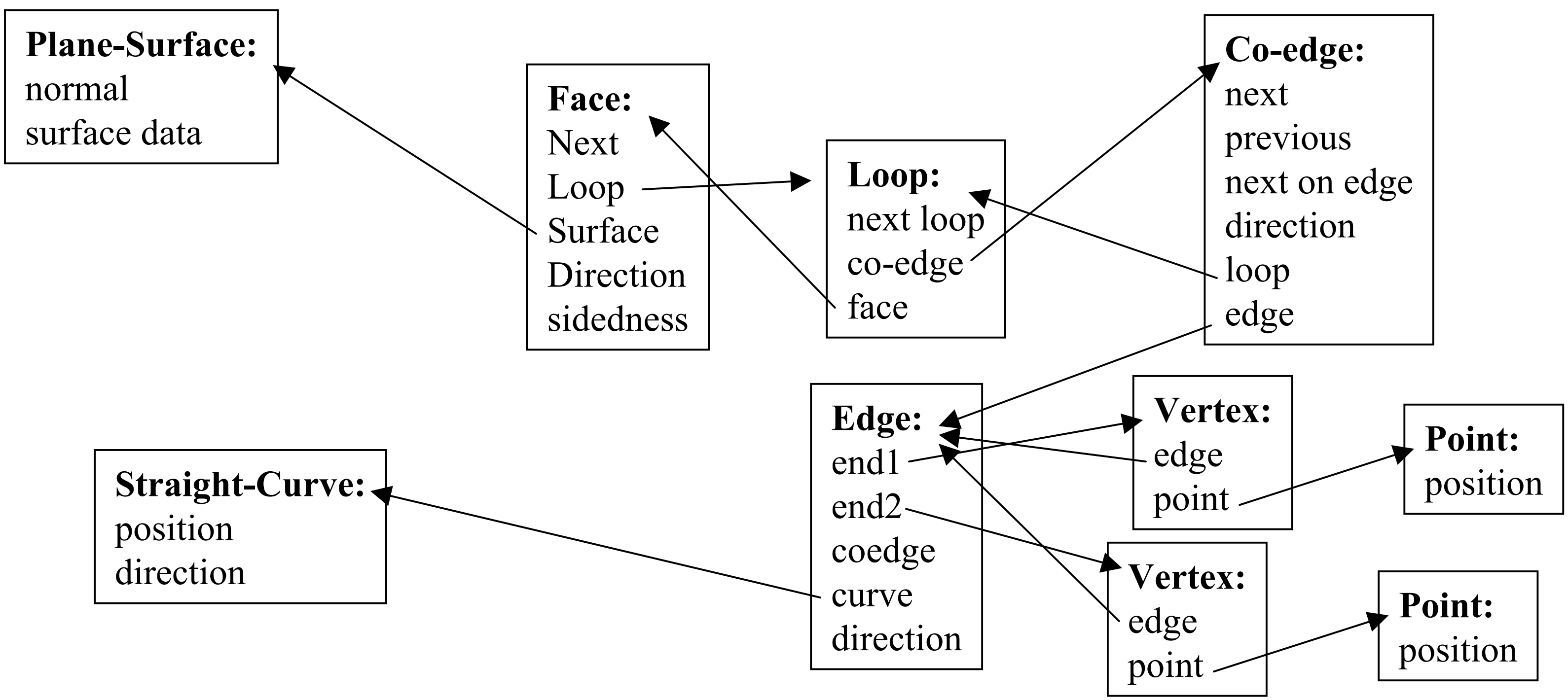
- 3D Model Shape Based Similarity Retrieval (Osada et al, Regli et al)
  - Uses D2 Distance measures
  - Works well for simple models
- Feature Based Model Retrieval (Regli et al)
  - Retrieves complete models
  - Feature interaction representation too simplistic
  - No method for indexing
- Group Technology
  - Goal is to group components by similar machining processes for improved factory flow
  - Similar machining processes does not guarantee shape similarity

Component Representation In the Database

- Data Represented in Three Layers
  - Raw B-Rep Component (Representation Layer)
  - Graph of Maximal Features and Interactions (Knowledge Layer)
  - Histograms of Features and Abstracted Interactions (Semantic Layer)
- Interactions Encoded at Multiple Abstraction Levels
  - Interaction Matrix
  - Set of Face Pairs
  - Abstracted Interaction

ACIS BREP Extraction

- ACIS is a robust geometric modeler whose output contains both Geometric and Topological data.
  - Too much information
  - Complexity of data interaction
    - Example a face:



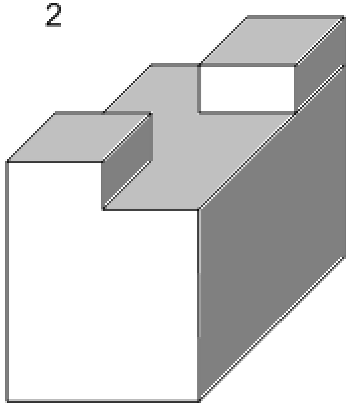
ACIS BREP Extraction

- Program written to extract data needed
  - Read in ACIS File
  - Parse Into Objects
  - Create BREP files in Following Format:

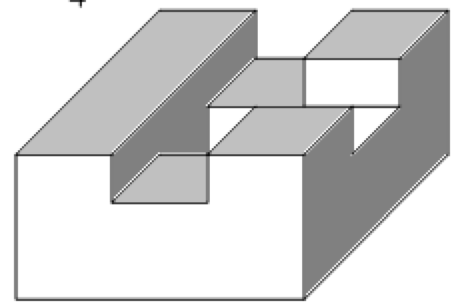
```
face 4
vertex 5 10 -20
vertex 5 20 -20
vertex 5 20 -5
vertex 5 10 -5

face 6
vertex -5 20 -20
vertex -5 10 -20
vertex -5 10 -5
vertex -5 20 -5

face 10
vertex -5 10 20
vertex -5 20 20
vertex -5 20 5
vertex -5 10 5

face 15
vertex 5 20 20
vertex 5 10 20
vertex 5 10 5
vertex 5 20 5

face 23
vertex 5 20 20
vertex 5 10 20
vertex -5 10 20
vertex -5 20 20
vertex -20 20 20
vertex -20 -20 20
vertex 20 -20 20
vertex 20 20 20
```

Demonstration of MakeBREP

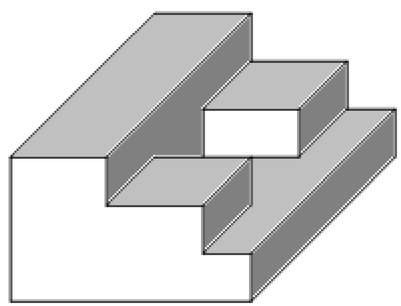
2



4



6



Feature Representation Overview

- Implementation is Based on “Constraint Based Feature Recognition” by Ming-Hsuan Yang and Dr. Michael Marefat
- Features are represented using a set of Predicates
- Basic Features: Pocket, Hole, Slot, Step, Blind-slot, Blind-step

Pre-defined Predicates

Table 1: Predefined predicates

Predicate	Semantics
Opp(f_1, f_2)	Two faces with opposite normals
Perp(f_1, f_2)	Two faces with perpendicular normals
Adj(f_1, f_2)	Two adjacent Faces
Type(f_1, R),Type(f_2, B)	Type of Faces: Real, Border
Concave(f_1, f_2)	Two adjacent faces that form a concave angle
Convex(f_1, f_2)	Two adjacent faces that form a convex angle
Unifiable(f_1, f_2)	Two faces that are unifiable
Num-Of-Opp-Face(f_1)	Number of Opp faces f_1 has
Num-Of-Real-Face(C_1)	Number of real faces of feature template C_1

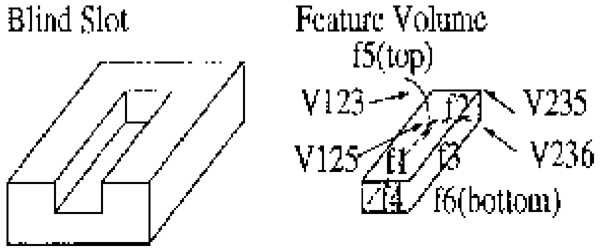


Figure 1: A blind slot

Feature Definition

- A feature C is a set of conjunctions of predicates within the geometric constraint language.

$C \leftarrow P_r \wedge P_b$

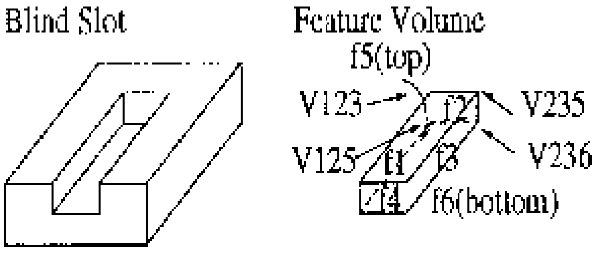


Figure 1: A blind slot

$$P_r = Opp(f_1, f_3) \wedge Perp(f_1, f_2) \wedge Perp(f_2, f_3) \wedge Perp(f_1, f_6) \wedge Perp(f_2, f_6) \wedge Perp(f_3, f_6) \wedge Type(f_1, R) \wedge Type(f_2, R) \wedge Type(f_3, R) \wedge Type(f_6, R) \wedge Adj(f_1, f_2) \wedge Adj(f_2, f_3) \wedge Adj(f_1, f_6) \wedge Adj(f_2, f_6) \wedge Adj(f_3, f_6) \wedge Concave(f_1, f_2) \wedge Concave(f_2, f_3) \wedge Concave(f_1, f_6) \wedge Concave(f_2, f_6) \wedge Concave(f_3, f_6)$$

$$P_b = Opp(f_2, f_4) \wedge Opp(f_5, f_6) \wedge Type(f_4, B) \wedge Type(f_5, B) \wedge Adj(f_1, f_4) \wedge Adj(f_3, f_4) \wedge Adj(f_1, f_5) \wedge Adj(f_2, f_5) \wedge Adj(f_3, f_5) \wedge Adj(f_4, f_5) \wedge Adj(f_4, f_6)$$

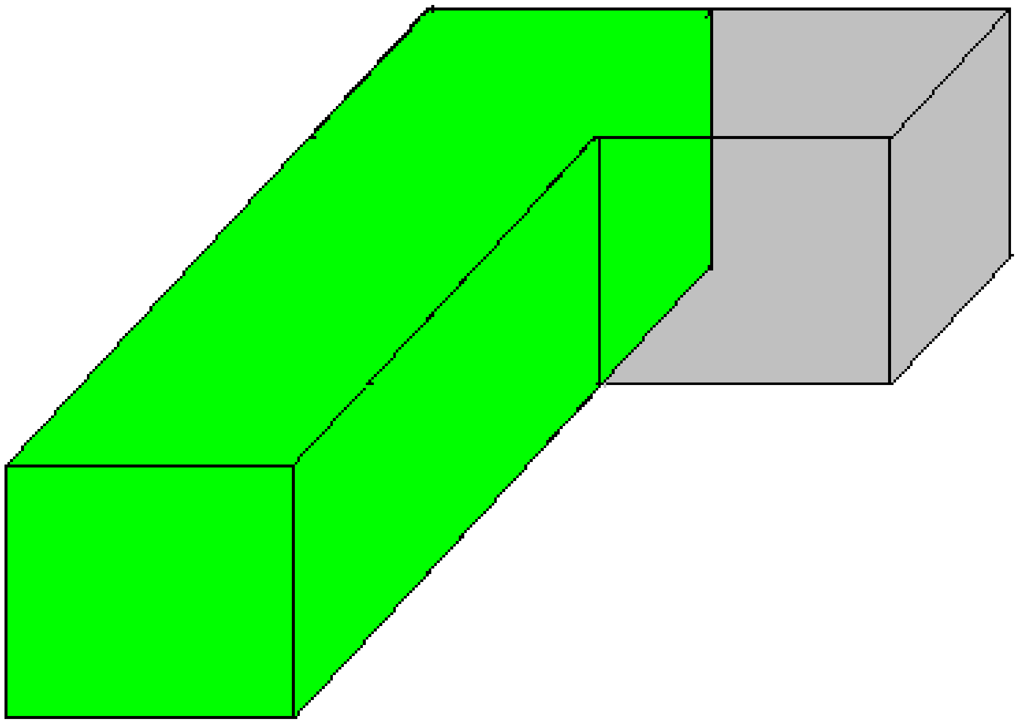
Feature Recognition Overview

- Geometric Constraint Satisfaction Feature Recognition
  - –Predicates for Canonical Features Known
  - –Predicates for Input Component are calculated
  - –Matching is performed by comparing Calculated Predicates against Canonical Feature Predicates
  - –Constraints are matches between feature predicates and component predicates

Feature Recognition Overview Continued…

- Constraints can be relaxed by ignoring it
  - Opposite face constraints can not be relaxed
  - Border and Real face type constraints can not be relaxed
  - Number of relaxations input by the user

Why do we need Relaxations



Practical Implementation

- MakePredicates.c
  - Needs Object file (BRE
  - and Delta file (BREP)
  - Identifies Border and Real faces
  - Generates normals for faces in Delta Volume
  - Defines predicates using normals and writes all these details to a file with extention .predicates

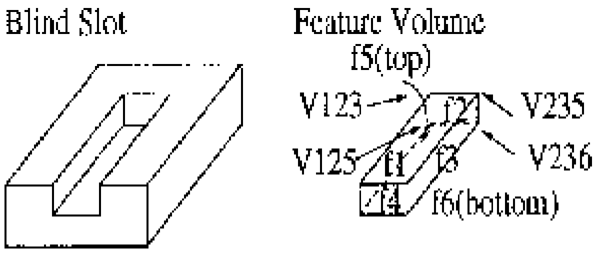


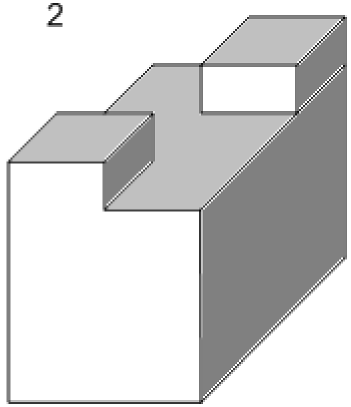
Figure 1: A blind slot

Predicates File Format

```
f b
f r
f r
f r
f b
f b
O 4 5
X 3 5
A 3 5
P 3 5
X 3 4
A 3 4
P 3 4
X 2 5
A 2 5
P 2 5
X 2 4
A 2 4
P 2 4
O 2 3
X 1 5
A 1 5
P 1 5
X 1 4
A 1 4
P 1 4
X 1 3
A 1 3
P 1 3
X 1 2
A 1 2
P 1 2
X 0 5
A 0 5
P 0 5
X 0 4
A 0 4
P 0 4
X 0 3
A 0 3
P 0 3
X 0 2
A 0 2
P 0 2
O 0 1
```

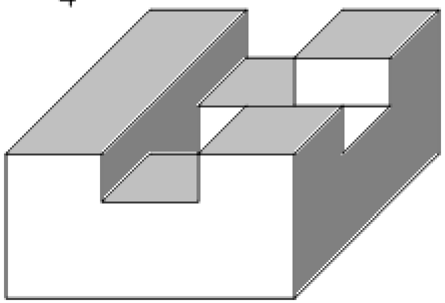
Demonstration of makePredicates.c

2



Part 2

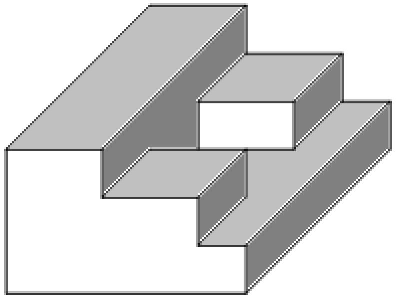
4



Part 4

Part 6

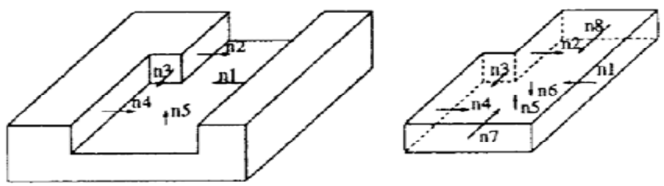
6

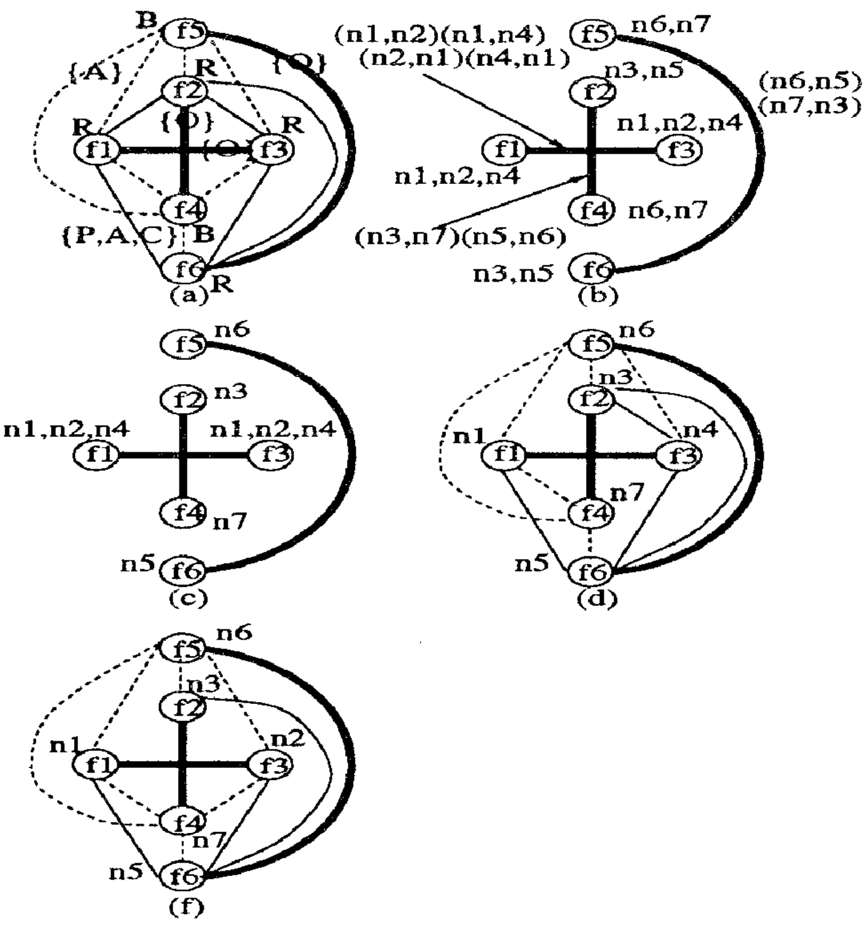


Practical Implementation of Feature Recognition

- featureRecognition.c
  - Requires
    - BREP files of Object and Delta Volume
    - Predicates of Component and canonical features
  - Identifies possible existences of Pocket, hole, blind slot, slot, blind step and step by
    - Making Nodes Consistent
    - Sorting on Cardinality
    - Making Arcs Consistent to find features and relaxing if necessary
    - Eliminating in consistent Features
  - Removing Duplicate features

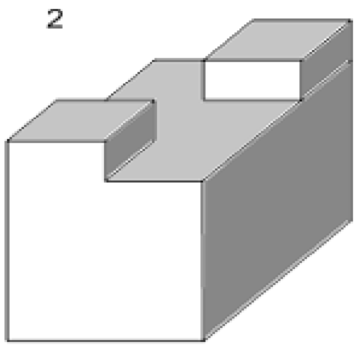
Constraint Graph

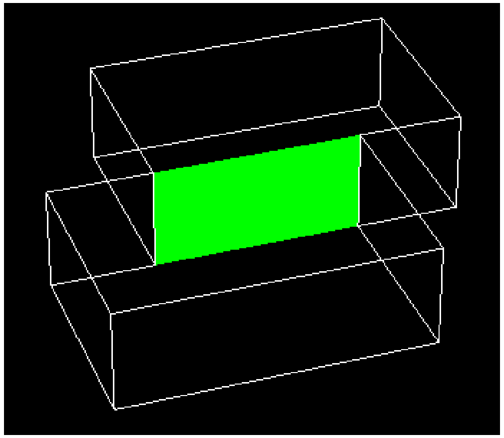


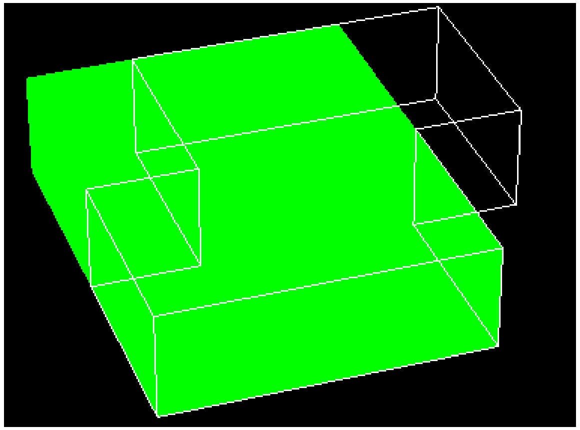


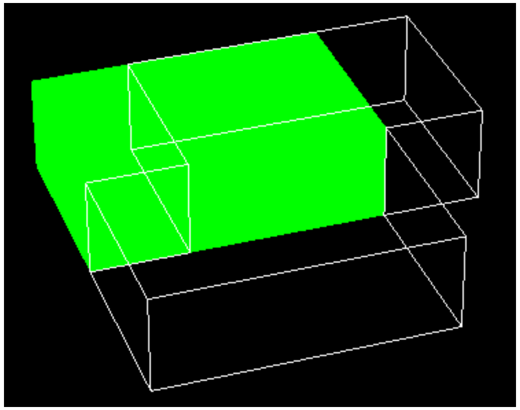
- Node Consistency
- Cardinality
- Arc Consistency

Inconsistent Features



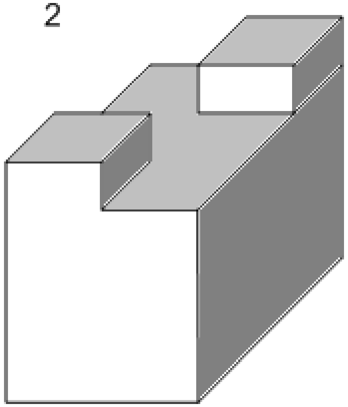






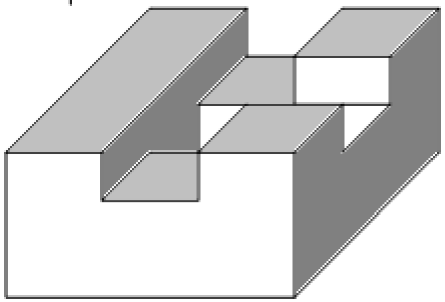
Demonstration of featureRecognition.c

2



Part 2

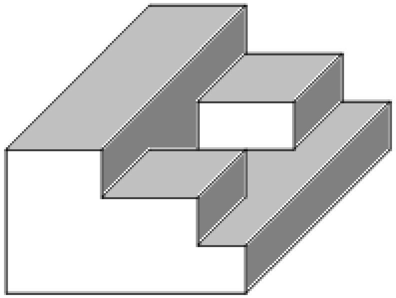
4



Part 7

Part 6

6

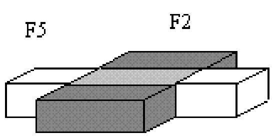


Feature Graph

- Generate graph of Features and Interactions
  - Nodes Represent Features
    - Maximum vertex
    - Minimum vertex
    - Face Information (Accessibility, Real or Border)
    - ID
    - Type
  - Arcs Represent Interactions
    - Starting Feature
    - Ending Feature
    - Matrix
    - Face Pairs
    - Type
    - ID

Interactions

- Interactions Represented as an nxm matrix where:
  - n is the number of faces in feature 1(f1)
  - m is the number of faces in feature 2 (f2)
  - Entries are in the set { +, -, s, i}
    - \+ indicates that f1 lies in the positive half space of f2
    - \- indicates that f1 lies in the negative half space of f2
    - s indicates they lie on the same plane
    - I indicates that the features interact



$R^{5,2}$

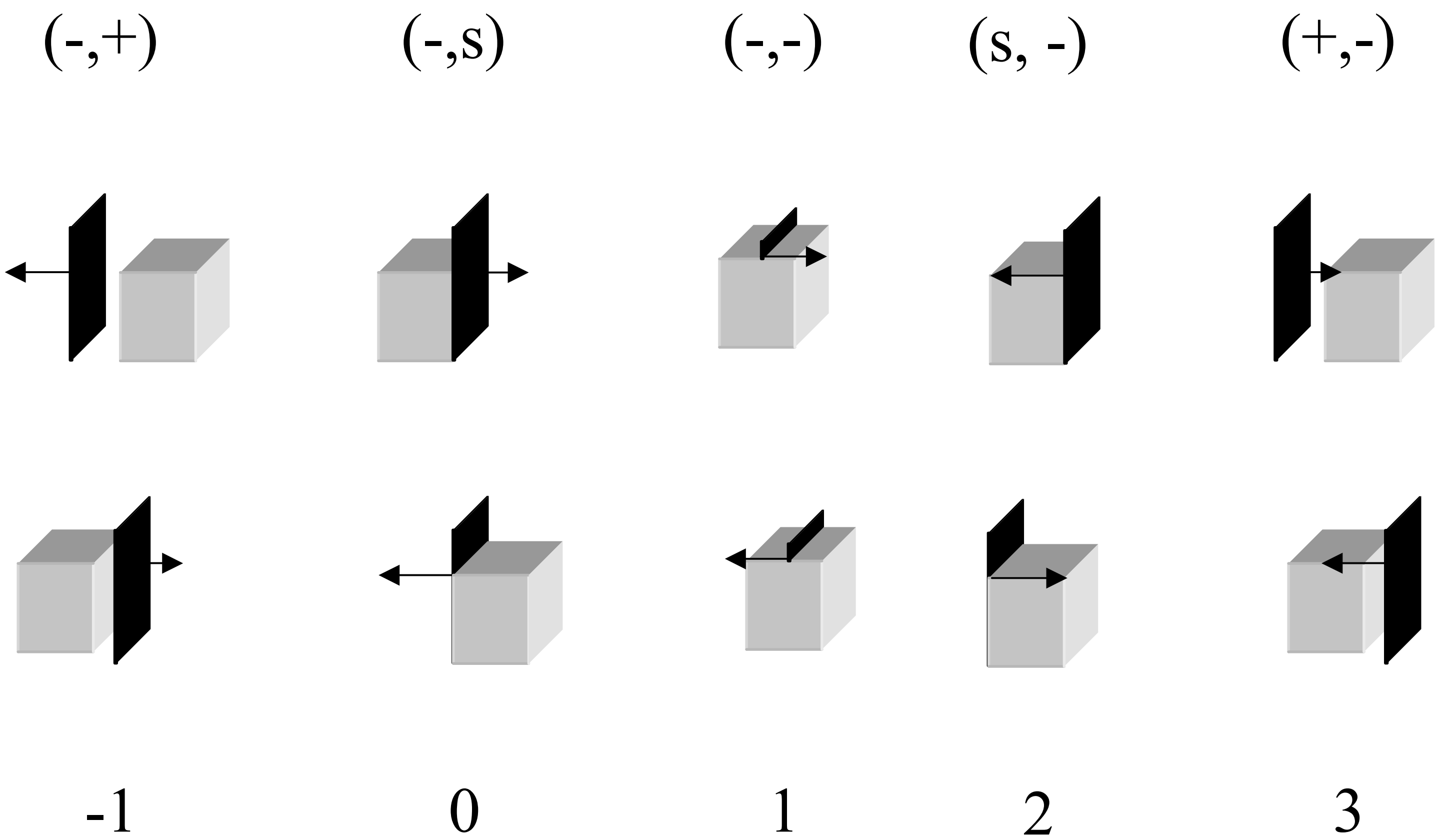
	1	2	3	4	5	6
1	s	-	i	i	-	-
2	-	s	i	i	-	-
3	-	-	+	-	-	-
4	-	-	-	+	-	-
5	-	-	i	i	-	-
6	-	-	i	i	-	-

Interactions Continued

- If only orthonormal features are considered
  - Results in 6x6 interaction Matrices
  - The near diagonal data points contain the pertinent data
  - The number of unique columns is reduced to 5 ordered types that are physically valid
    - The following notation indicates (same, opposite) face
    - (-,+) No Interaction
    - (-,s) indicates a shared face with no internally shared points
    - (-,-) indicates a face that is interior to both faces of the other feature
    - (s,-) indicates a shared face with internally shared points
    - (+,-) indicates a face that is not interior to one face of the other feature
  - The following Combinations are invalid:
    - (+,+) can not be outside both faces parallel to the face of interest
    - (s,s) can not be the same as two faces which bound a feature
    - (+,s) and (s,+) can not be same as one face and outside the other

Interactions Continued

Note: Arrows Point to interior of second feature



Interactions Continued

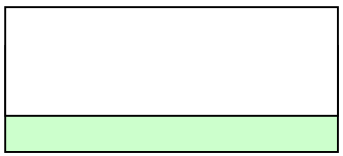
- Pairs of parallel Faces are compared Resulting in 7 Feasible Unordered face-pair Combinations:

Type	Face 1	Face 2	Relation of faces to other Feature
D	(s,-)	(s,-)	Both Faces Shared
B	(s,-)	(-,-)	One Face Shared Other Inside
E	(s,-)	(+,-)	One face shared, Other Outside Opp
	(s,-)	(-,s)	Not Possible
A	(-,-)	(-,-)	Both Faces Inside
C	(-,-)	(+,-)	One Face Inside, Other Outside
	(-,-)	(-,s)	Not Possible
F	(+,-)	(+,-)	Both Outside
G	(+,-)	(-,s)	One Face Shared, Other Outside
	(-,s)	(-,s)	Not Possible

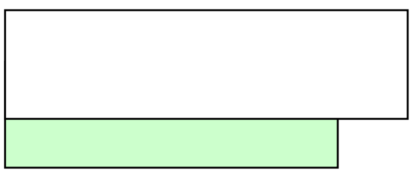
Interactions Continued

- Inverse Relations Exist for some Face-pair Types.

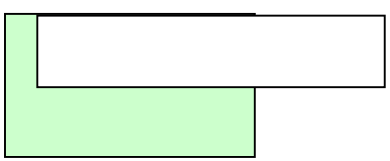
Type	Face 1	Face 2	Inverse Interaction
D	(s,-)	(s,-)	D
B	(s,-)	(-,-)	E
E	(s,-)	(+,-)	B
A	(-,-)	(-,-)	F
C	(-,-)	(+,-)	C
F	(+,-)	(+,-)	A
G	(+,-)	(-,s)	G



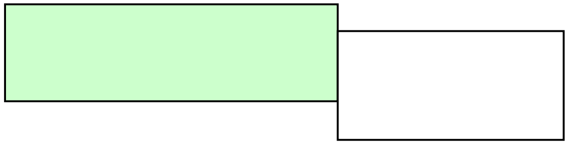
D



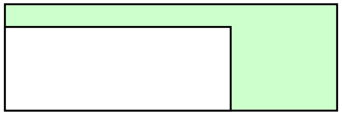
E



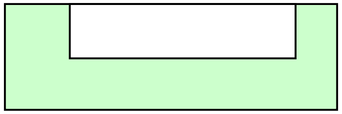
C



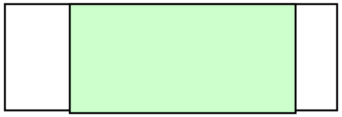
G



B



A



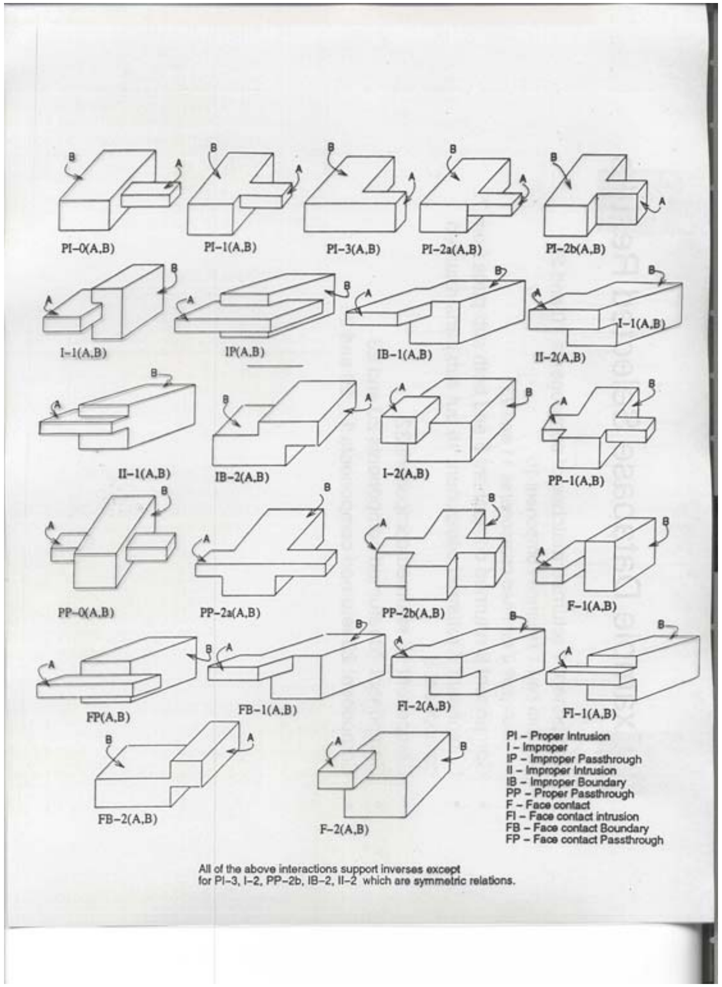
F

Interactions Continued

- Several Face-pair types indicate a family of interactions.

Type	Face 1	Face 2	Interaction Family
E	-s	-+	Intrusion
C	--	-+	Improper Interaction
F	-+	-+	Pass Through
G	-+	+s	Face

Show All Interactions Here



Interactions Continued

Each Interaction can be represented by three Face-pairs

	Normal			Inverse		
pi-0	A	A	E	F	F	B
pp-0	A	A	F	F	F	A
pi-1	A	B	E	F	E	B
pp-1	A	B	F	F	E	A
l-1	A	C	C	F	C	C
ii-1	A	C	E	F	C	B
f-1	A	C	G	F	C	G
ip	A	C	F	F	C	A
pi-2b	A	D	E	F	D	B
pp-2b	A	D	F	F	D	A
fi-1	A	E	G	F	B	G
fp	A	F	G	F	A	G
pi-2a	B	B	E	E	E	B
pp-2a	B	B	F	E	E	A
ib-1	B	C	C	E	C	C
fb-1	B	C	G	E	C	G
fi-2	B	E	G	E	B	G
l-2	C	C	C	C	C	C
ib-2	C	C	D	C	C	D
f-2	C	C	G	C	C	G

Feature Graph Creation Algorithm

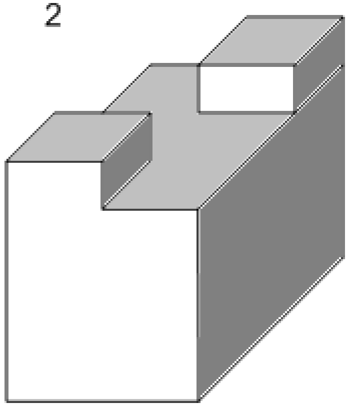
- For feature in component:
  - Create Node
- For Node1 in component
  - For Node2 with ID Greater than Node1
    - If Node1 interacts with Node2
      - Compare Faces and Generate Matrix and Face Pairs
      - Determine Face Type

Maximal Feature Sub-Graph Extraction

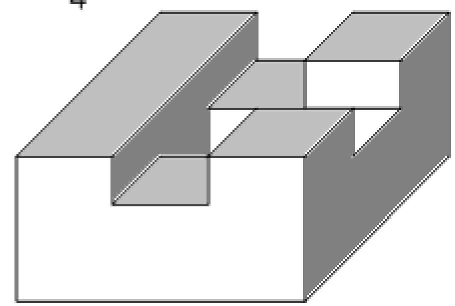
- For Node in Component Graph
  - If Node is subsumed (no 3s) or
  - If Node Ends inside another feature
    - Delete Node
    - For Arc in Component Graph
      - If arc associated with Node
        - Delete Arc

Demonstrate msfgExtract

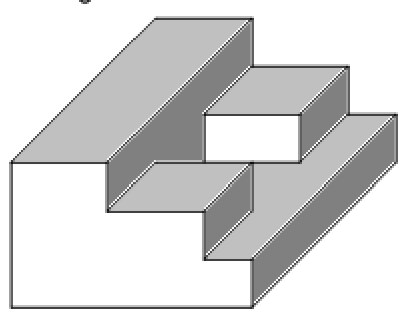
2



4



6



Future Work

- Block Extractor
- Generate Database of Parts
  - MySQL to be used
- Query Relaxation Unit
  - Two methods being considered
- Query Evaluation Unit
  - Perform graph matching on the candidate matches
- Implement Spatial Logic Unit
  - Complexity Dependent on Query Relaxation Technique