LECTURE # 8

Solid Modeling

Assembly Modeling

Solid Modeling

In the solid modeling, the solid definitions include vertices (nodes), edges, surfaces, weight, and volume. The model is a complete and unambiguous representation of a precisely enclosed and filled volume.

Advantages:

- Has all the advantages of surface models (uniqueness, non-ambiguous, realistic, surface profile) plus volumetric information.
- Allows the designer to create multiple options for a design.
- 2D standard drawings, assembly drawing and exploded views are generated form the 3D model.

Solid Modeling

Advantages:

- Can easily be exported to different FEM programs for analysis.
- Can be used in newly manufacturing techniques;
  - CIM
  - CAM
  - DFM, DFA
- Mass and volumetric properties of an object can be easily obtained; total mass, mass center, area and mass moment of inertia, volume, radius of gyration, …

Solid Modeling

Disadvantages:

- More intensive computation than wireframe and surface modeling.
- Requires more powerful computers (faster with more memory and good graphics), not a problem any more.

Historical Background

CNC: ~1950

Mainframe Computers: ~1960's

Boundary Representation: 1970
(Bruce Baumgart, Ian C. Braid)

Constructive Solid Geometry: 1974
(Ian C. Braid, Ari Requicha, John Woodwark)

Geometry

Geometric data relates to dimensional information, e.g.

– the location of points in space

– the shape and size of geometric features.

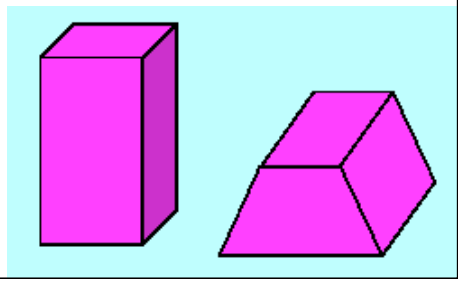
Topology

Topology: refers to the connectivity of the elements which make up the model, e.g.

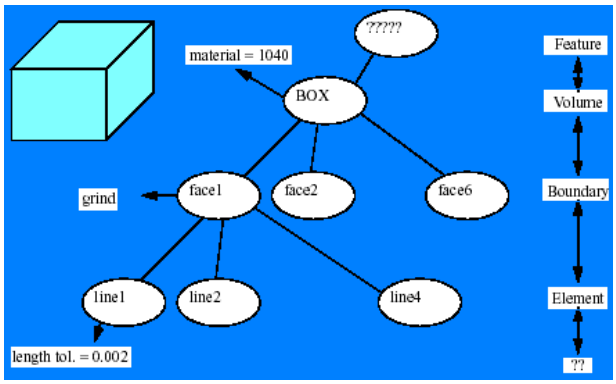
– two faces intersect at an edge

Inclusion of topological data makes solid models *computationally verifiable.*

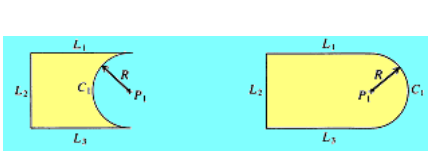
Two objects with the same topology but different geometry.



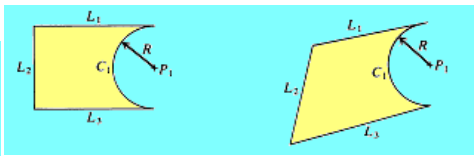
Solid Modeling



Topology



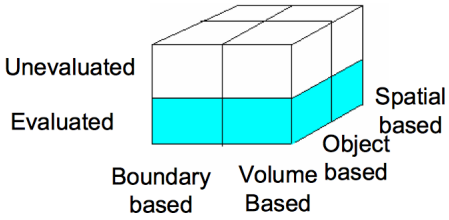
Same Geometry,
Different Topology



Different Geometry,
Same Topology

Classification of Solid Modeling

- Boundary based or Volume based
- Object based or spatially based
- Evaluated or unevaluated



		Boundary |	Volume
Unevaluated	Spatial	Half space	Octree
	Object	Euler Ops	CSG
Evaluated	**Spatial**	**Boundary Cell Enum**	**Cell Enum**
	Object	**Boundary Rep**	**Non-parametic**

Solid representation schemes

1. Half Spaces
2. Boundary Representation (B-rep)
3. Constructive Solid Geometry (CSG)
4. Sweeping
5. Analytical Solid Modeling (ASM)
6. Cell decomposition
7. Spatial Ennumeration
8. Octree encoding
9. Primitive Instancing

The 3 most popular schemes : B-rep, CSG, Sweeping

Some Solid Modelers in Practice

Modeler	Developer	Primary Scheme	User Input
CATIA	IBM	CSG	BREP+CSG
GEOMOD / I-DEAS	SDRC/EDS	BREP	BREP+CSG
PATRAN-G	PDA ENGG.	ASM	HYPERPATCHES+CSG
PADL-2	CORNELL UNI.	CSG	CSG
SOLIDESIGN	COMPUTER VISION	BREP	BREP+CSG
UNISOLIDS / UNIGRAPHICS	McDONELL DOUGLAS	CSG	BREP+CSG
PRO-E	PARAMETRIC	BREP	BREP+CSG
SOL. MOD. SYS	INTERGRAPH	BREP	BREP+CSG

1

Half Spaces

Half Spaces

Half space from a basic representation scheme for bounded solids. A half space is a regular point set in E^3 and is given by :

$$H = \{P : P \in E^3 \text{ and } f(P) < 0\}$$

Advantages :
The main advantage is its conciseness of representation compared to other modeling schemes.
It is the lowest representation available for modeling a solid object.

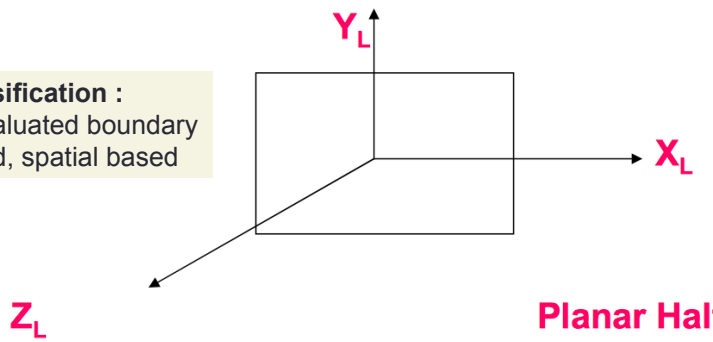
Disadvantages:
The representation can lead to unbounded solid models as it depends on user manipulation of half spaces.
The modeling scheme is cumbersome for ordinary users/ designers.

Half Spaces

A planar half surface is represented as :

$$H = \{(x, y, z) : z < 0\}$$

Classification :
unevaluated boundary based, spatial based

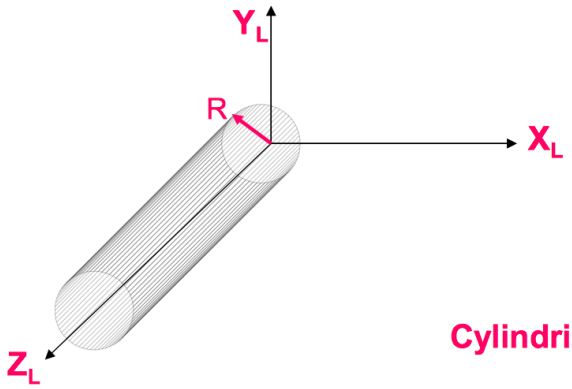


Planar Half Space

Half Spaces

A cylindrical half space is a given by :

$$H = \{(x, y, z) : x^2 + y^2 < R^2\}$$

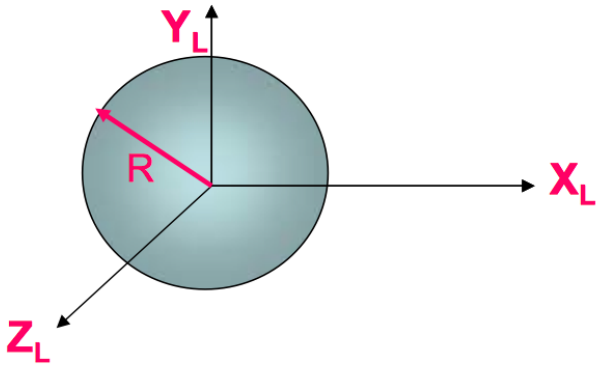


Cylindrical Half Space

Half Spaces

A spherical half space is a given by:

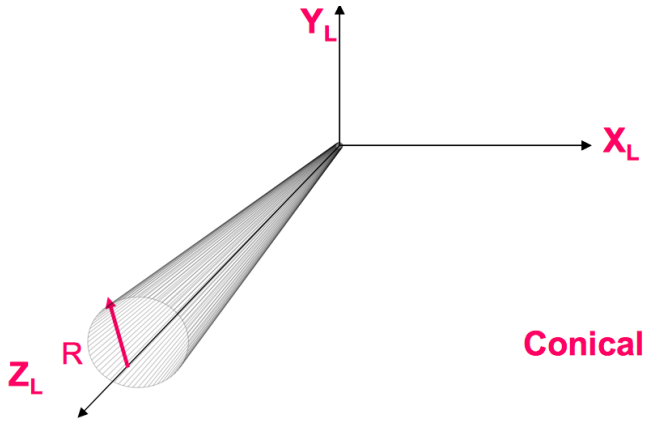
$$H = \{(x, y, z) : x^2 + y^2 + z^2 < R^2\}$$



Half Spaces

A conical half space is a given by :

$$H = \{(x, y, z) : x^2 + y^2 < (\tan(\alpha / 2)z)^2\}$$

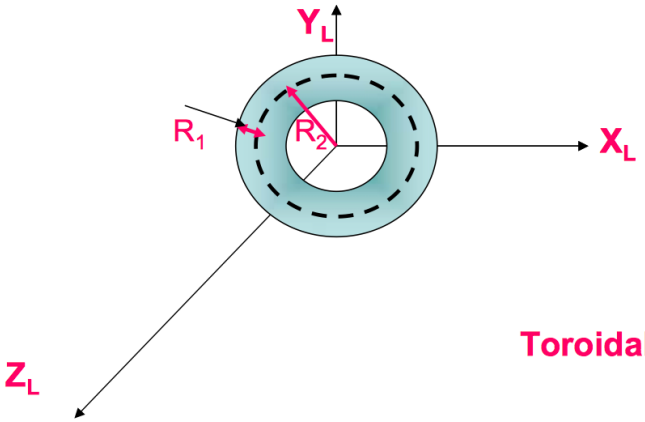


Conical Half Space

Half Spaces

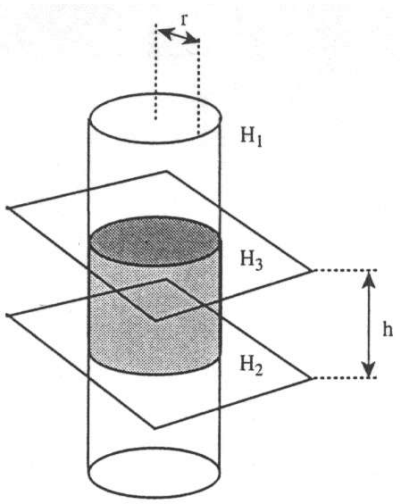
A torodial half space is a given by :

$$H = \{(x, y, z) : (x^2 + y^2 + z^2 - R_2^2 - R_1^2 < 4R_2^2(R_1^2 - z^2)\}$$



Toroidal Half Space

Half Spaces



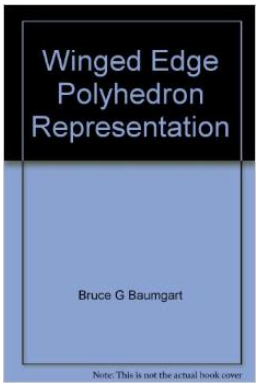
2

Boundary Representation

B-rep

Boundary Representation

The basic method for B-rep was developed independently in the early 1970s by both Ian C. Braid in Cambridge (for CAD) and Bruce G. Baumgart at Stanford (for computer vision).



I.C.Braid, I.C.Hillyard & I.A. Stroud, Stepwise construction of polyhedra in geometric modeling, in K.W.Brodie, ed. *Mathematical Methods in Computer Graphics & Design,* Academic Press (1980) 123-141.

B.G. Baumgart, Winged edge polyedron representation, Stanford University, Stanford, CA, 1972.

Boundary Representation (B-rep)

Closed surface : One that is continuous without breaks

Oriented surface : One in which is possible to distinguish two sides by using surface normal to point to the inside or outside of the solid under consideration.

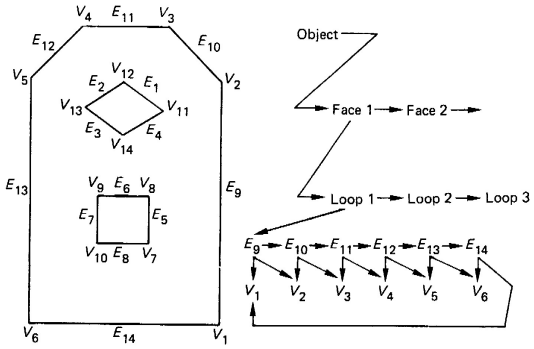
Boundary model : Boundary model of an object is comprised of closed and orientable faces, edges and vertices. A data base of a boundary model contains both its topology and geometry.

Topology: Created by Euler operations

Geometry: includes coordinates of vertices, rigid motions and transformations.

Boundary representation (B-rep)

In a boundary representation (B-rep) modeller, a solid is represented by an evaluated data structure containing the elements which describe its boundary.



B-Reps for Solids

Together they constitute the *combinatorial structure* of the representation.

The vertex coordinates are the *metric information* associated with the representation.

In the geometric modeling jargon, the combinatorial structure is often called the *topology*, and the metric information the *geometry*.

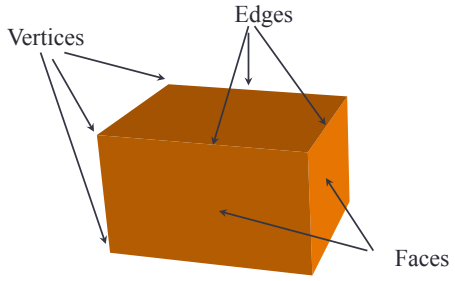
B-Rep vs. Surface Modeling

- **Surface model:**
  – A collection of surface entities which simply enclose a volume lacks the connective data to define a solid (i.e. topology).

- **B-Rep model:**
  – Technique guarantees that surfaces definitively divide model space into solid and void, even after model modification commands.

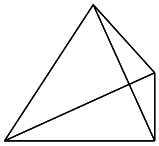
Boundary Representation (B-rep)

- Solids represented by faces, edges and vertices
- Topological rules must be satisfied to ensure valid objects
  - faces bounded by loop of edges
  - each edge shared by exactly two faces
  - each edge has a vertex at each end
  - at least 3 edges meet at each vertex



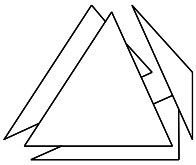
Boundary Representation (B-rep)

Boundary of a solid…



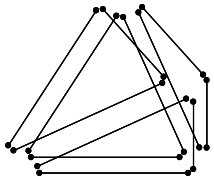
(a) Solid: bounded, connected subset of E^3

Boundary of surfaces…



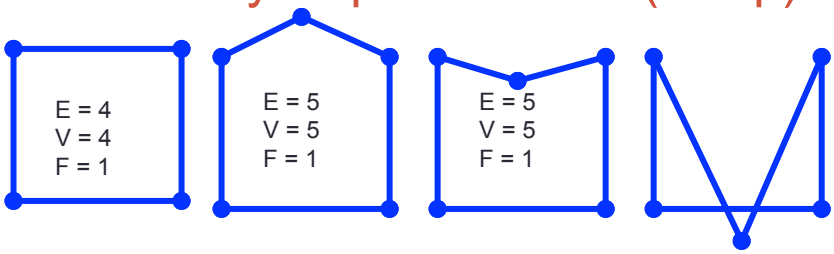
(b) Faces: boundary of solid
bounded, connected subsets of Surfaces

Boundary of curves (edges)…



(c) Edges: boundary of faces
bounded, connected subsets of curves

Boundary Representation (B-rep)



Original object Modified object Nonsense object

Euler Operations (Euler-Poincare Law)
The validity of resulting solids is ensured via Euler operations which can be built into CAD/CAM systems.

Volumetric Property calculation in B-rep:
It is possible to compute volumetric properties such as mass properties by virtue of Gauss divergence theorem which converts volume integrals to surface integrals.

Winged Edge data structure

- It is a data representation used to describe polygon models in computer graphics.
- Baumgart's *winged-edge* data structure is the oldest data structure for a B-rep.
- It is quite different from that of a wireframe model, because the winged-edge data structure uses *edges* to keep track almost everything.
- Topologically, one can always stretch curvilinear edges and faces so that they become flat without changing the relationships among them.

Winged Edge data structure

For each edge, the following information are important:

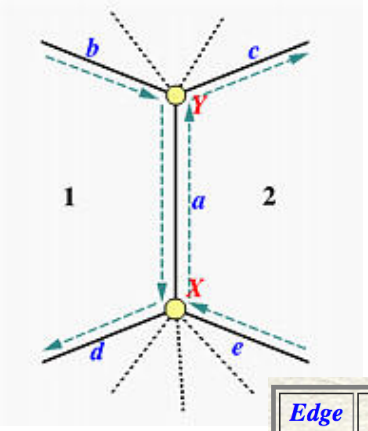
- vertices of this edge,
- its *left* and *right* faces,
- the predecessor and successor of this edge when traversing its left face, and
- the predecessor and successor of this edge when traversing its right face.

Winged Edge data structure

The Edge Table
Each entry in the edge table contains those information :

- edge name,
- start vertex and end vertex,
- left face and right face,
- the predecessor and successor edges when traversing its left face,
- the predecessor and successor edges when traversing its right face.

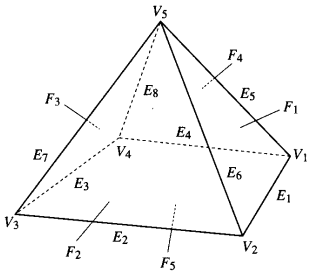
Winged Edge data structure



The four edges **b**, **c**, **d** and **e** are the *wings* of edge **a** and hence edge **a** is "winged."

Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	X	Y	1	2	b	d	e	c

B-Rep Data Structure

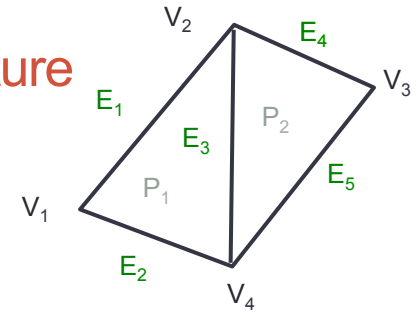


Face Table	
Face	*Edges*
F_1	E_1, E_5, E_6
F_2	E_2, E_6, E_7
F_3	E_3, E_7, E_8
F_4	E_4, E_8, E_5
F_5	E_1, E_2, E_3, E_4

Edge Table	
Edge	*Vertices*
E_1	V_1, V_2
E_2	V_2, V_3
E_3	V_3, V_4
E_4	V_4, V_1
E_5	V_1, V_5
E_6	V_2, V_5
E_7	V_3, V_5
E_8	V_4, V_5

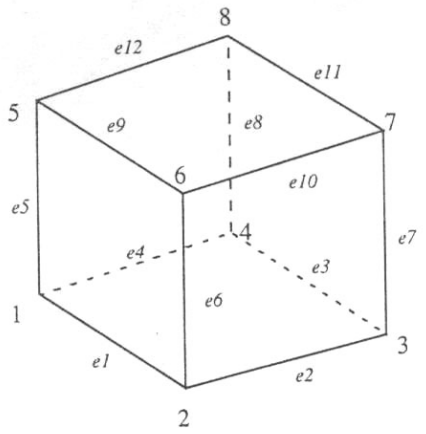
Vertex Table	
Vertex	*Coordinates*
V_1	x_1, y_1, z_1
V_2	x_2, y_2, z_2
V_3	x_3, y_3, z_3
V_4	x_4, y_4, z_4
V_5	x_5, y_5, z_5
V_6	x_6, y_6, z_6

B-Rep Data Structure



Vertex #	Location	Edge #	Vertices	Polygon #	Edges
V_1	0,5,0	E_1	V_1 , V_2	P_1	E_1 , E_3 , E_2
V_2	4,15,0	E_2	V_1 , V_4	P_2	E_3 , E_4 , E_5
V_3	4,10,0	E_3	V_2 , V_4		
V_4	8,0,0	E_4	V_2 , V_3		
		E_5	V_3 , V_4		

Vertex-based B-rep



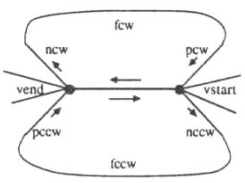

```
v1 x1 y1 z1
v2 x2 y2 z2
v3 x3 y3 z3
v4 x4 y4 z4
v5 x5 y5 z5
v6 x6 y6 z6
v7 x7 y7 z7
v8 x8 y8 z8
```

```
f1 v1 v2 v3 v4
f2 v6 v2 v1 v5
f3 v7 v3 v2 v6
f4 v8 v4 v3 v7
f5 v5 v1 v4 v8
f6 v8 v7 v6 v5
```

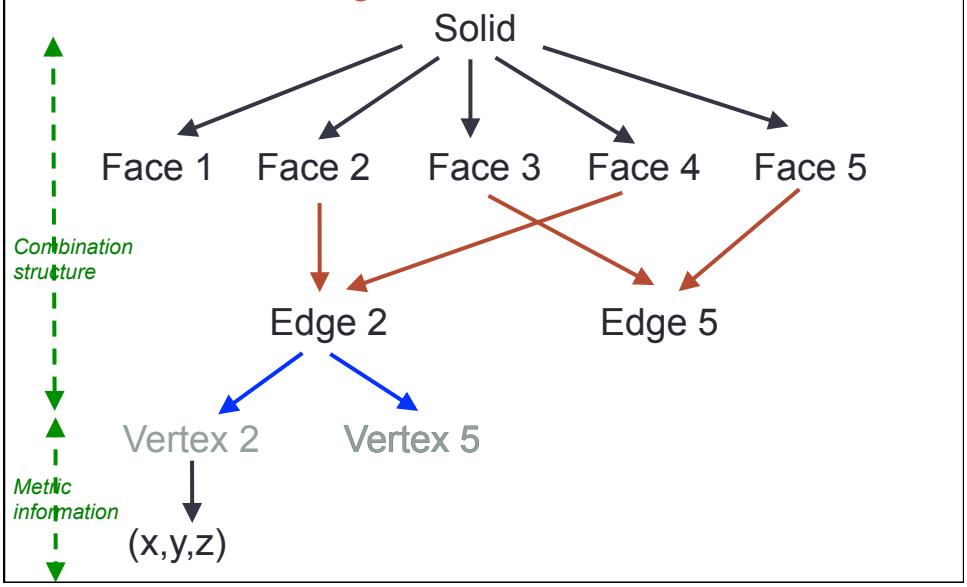
Winged-edge data structure

eid	vstart	vend	fcw	fccw	ncw	pcw	nccw	pccw
e1	v1	v2	f1	f2	e2	e4	e5	e6
e2	v2	v3	f1	f3	e3	e1	e6	e7
e3	v3	v4	f1	f4	e4	e2	e7	e8
e4	v4	v1	f1	f5	e1	e3	e8	e5
...								...
e12	v8	v5	f5	f6	e5	e8	e11	e9

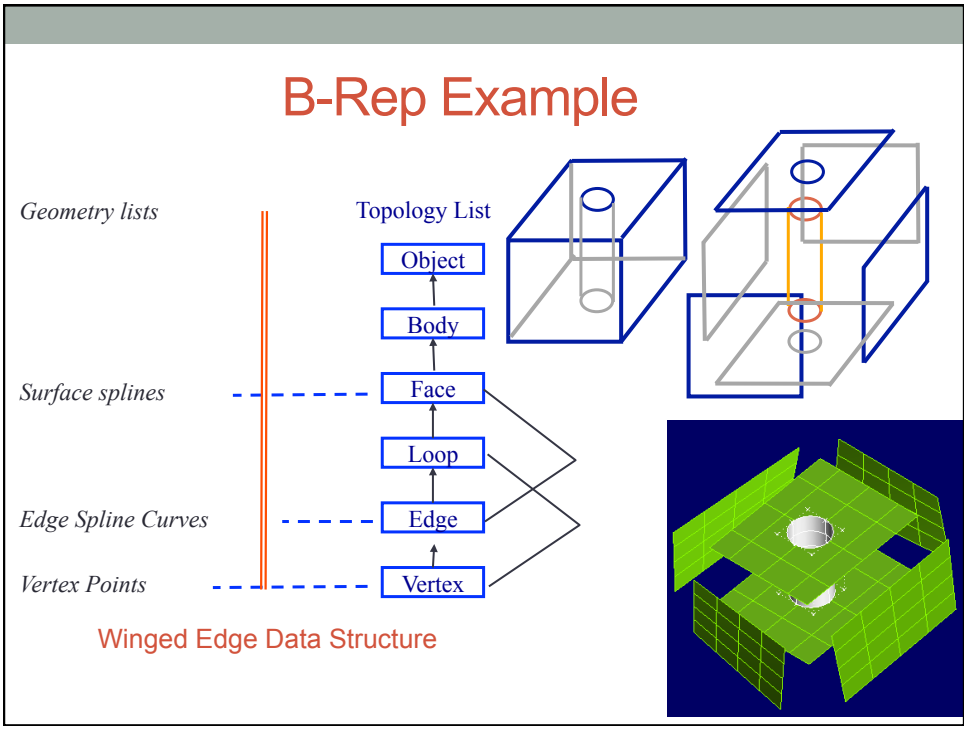
vid	estart	coords	fid	estart
v1	e1	x1 y1 z1	f1	e1
v2	e2	x2 y2 z2	f2	e9
...			...	
v8	e12	x8 y8 z8	f6	e9



A pyramid is represented by a graph containing face, edge and vertex nodes.

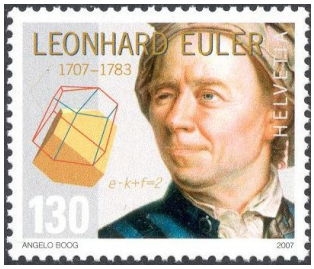


B-Rep Example



Winged Edge Data Structure

Euler-Poincare Formula



Leonard Euler (1707-1783)

Henri Poincaré (1854-1912)

Euler (1752) a Swiss mathematician proved that polyhedra that are homomorphic to a sphere are topologically valid if they satisfy the equation.

Euler's Law

Defines an invariant relationship among the vertices, edges, and face loops of a polyhedron.

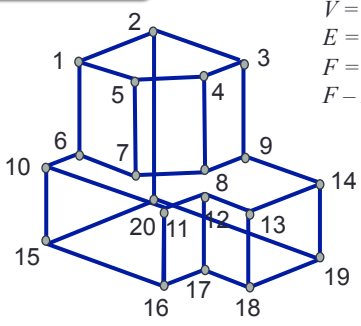
Valid for simple polyhedra (continuous, no hole)

V - E + F = 2

V (# vertices)

E (# edges)

F (# face loops)



$V = 20$
$E = 30$
$F = 12$
$F - E + V = 2$

Expanded Euler's Law

- For evaluating complex polyhedrons formulation is expanded to include:
  - hole loops, (any loop which is completely enclosed within another)
  - through holes or genus (a feature that completely penetrates the object adds to its genus, no penetrating features, genus = 0)
  - shells (sets of faces which bound a volume, either space or void)

Expanded Euler's Law

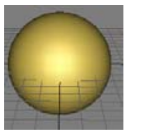
This is known as the Euler-Poincare Law:

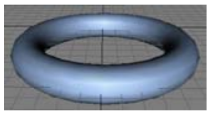
$$V - E + F - L = 2(S - G)$$

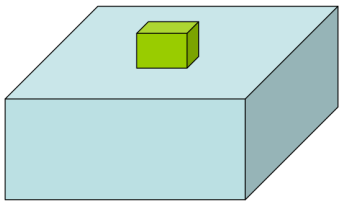
V = # of vortives
E= # of edges
F = # of faces
L = # of hole loops
(sometimes "H" for hole)
S = # of shell bodies
(sometimes "C")
G = # of thru holes, genus, passage features

Loops (rings), Genus & Bodies

- **Genus zero**
- **Genus one**
- **Genus two**

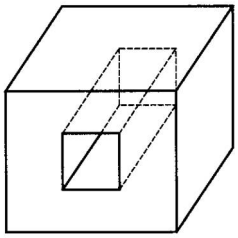




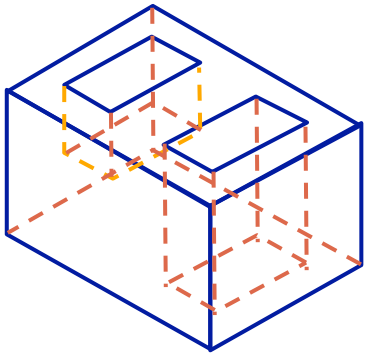


- **One inner loop**

Euler-Poincare Formula



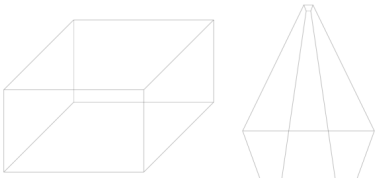
$16 - 24 + 10 - 2 = 2(1 - 1)$



V - E + F - L = 2 (S – G)
24 - 36 + 15 - 3 = 2 (1 - 1)

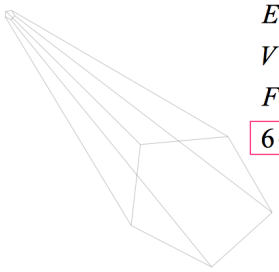
Validity Checking for Simple Solids

$F - E + V = 2$ Simple Solids

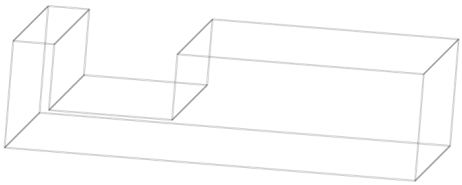


$E = 12$
$V = 8$
$F = 6$
$6 - 12 + 8 = 2$

$E = 8$
$V = 5$
$F = 5$
$5 - 8 + 5 = 2$



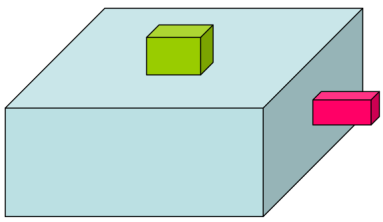
$E = 10$
$V = 6$
$F = 6$
$6 - 10 + 6 = 2$



$E = 24$
$V = 16$
$F = 10$
$10 - 24 + 16 = 2$

Validity Checking for Polyhedra with inner loops

$F - E + V - L = 2(B - G)$ General

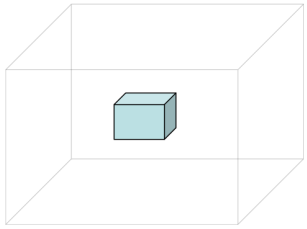


$E = 36$
$F = 16$
$V = 24$
$L = 2$
$B = 1$
$G = 0$
$16 - 36 + 24 - 2 = 2(1 - 0) = 2$

Validity Checking for Polyhedra with holes

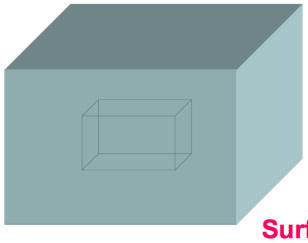
$$F - E + V - L = 2(B - G) \qquad \text{General}$$

Interior hole (void)



$E = 24$
$F = 12$
$V = 16$
$L = 0$
$B = 2$
$G = 0$

$12 - 24 + 16 - 0 = 2(2 - 0) = 4$



$E = 24 \qquad F = 11$
$V = 16 \qquad L = 1$
$B = 1 \qquad G = 0$

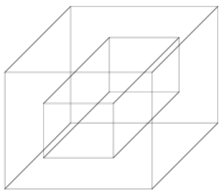
$11 - 24 + 16 - 1 = 2(1 - 0) = 2$

Surface hole

Validity Checking for Polyhedra with through holes (handles)

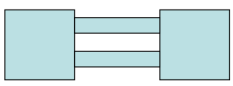
$$F - E + V - L = 2(B - G) \qquad \text{General}$$

Through hole



$E = 24$
$F = 10$
$V = 16$
$L = 2$
$B = 1$
$G = 1$

$10 - 24 + 16 - 2 = 2(1 - 1) = 0$



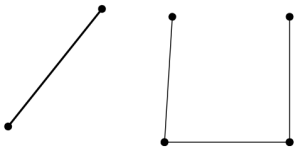
$E = 48 \quad F = 20$
$V = 32 \qquad L = 4$
$B = 1 \qquad G = 1$

$20 - 48 + 32 - 4 = 2(1 - 1) = 0$

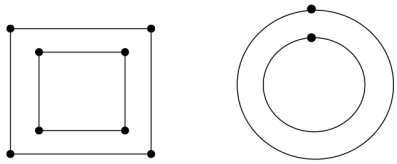
Handles/through hole

Validity Checking for open objects

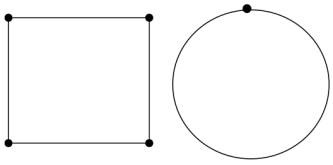
$$F - E + V - L = B - G$$



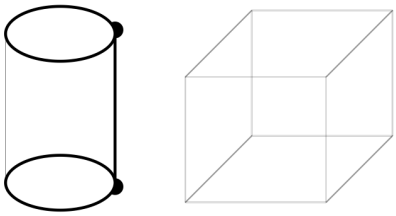
Wireframe polyhedra



Shell polyhedra



Lamina polyhedra



Open three dimensional polyhedra

Euler Operators

Geometric entities stored in B-Rep data structures are the *shell, face, loop, edge,* and *vertex*.

Operators are needed to manipulate these entities (e.g., an operator to make an edge, an operator to delete an edge,…)

The process that adds and deletes these boundary components is called an Euler operation.

Euler Operators

A connected structure of vertices, edges and faces that always satisfies Euler's formula is known as Euler object.

Applicability of Euler formula to solid objects:

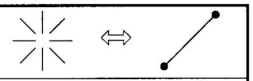
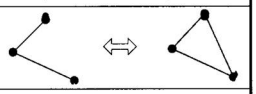
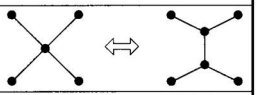
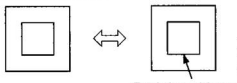
- At least three edges must meet at each vertex.
- Each edge must share two and only two faces
- All faces must be simply connected (homomorphic to disk) with no holes and bounded by single ring of edges.
- The solid must be simply connected with no through holes

Euler Operators

- Model is topologically valid if satisfies Euler-Poincare
- Validity relationship allows for definition of a set of operators:
  - known as *Euler Operators*
  - allow faces, edges and vertices to be added and removed from model while retaining validity

	v	e	f	g	s	l	
mvfs	1	0	1	0	1	0	Make vertx, face, body
mev	1	1	0	0	0	0	Make edge, vertex
mef	0	1	1	0	0	0	Make edge, face
kemr	0	-1	0	0	0	1	Kill edge, make ring
kev	-1	-1	0	0	0	0	Kill edge, vertex
kef	0	-1	-1	0	0	0	Kill edge, face
mekr	0	1	0	0	0	-1	Make edge, kill ring

Euler Operators

MEVVLS (KEVVLS)	make (kill) edge, two vertices, loop, shell	
MEL (KEL)	make (kill) edge, loop	
MEV (KEV)	make (kill) edge, vertex	
MVE (KVE)	make (kill) vertex, edge	
MEKH (KEMH)	make (kill) edge, kill (make) hole	
MZEV (KZEV)	make (kill) zero length edge, vertex	
MPKH (KPMH)	make(kill) peripheral loop, kill (make) hole loop	

3

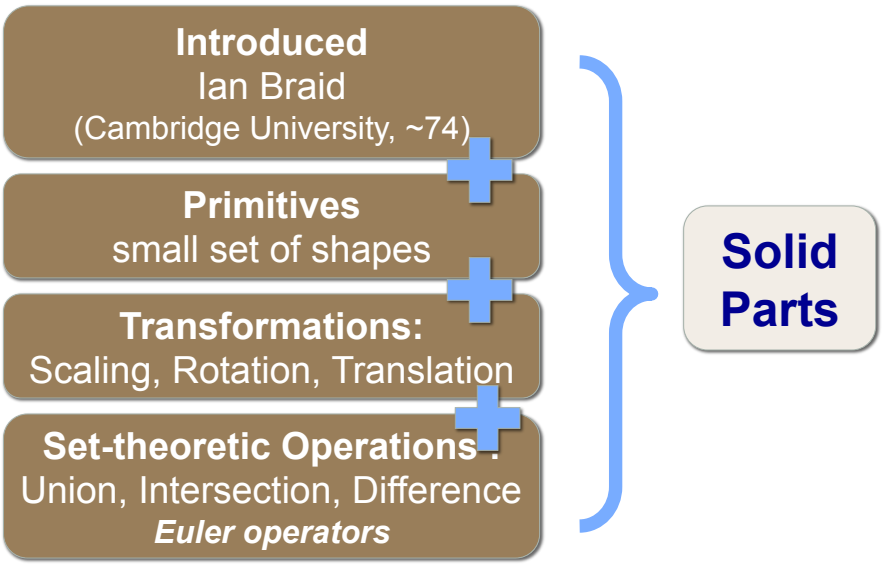
Constructive Solid Geometry - CSG

Constructive Solid Geometry, CSG

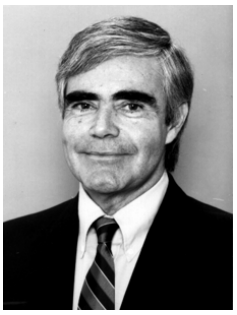
- CSG defines a model in terms of combining basic and generated (using extrusion and sweeping operation) solid shapes.
- CSG uses Boolean operations to construct a model (George Boole, 1815-1864, invented Boolean algebra).
- There are three basic Boolean operations:

Union **(Unite, join) - the operation combines two volumes included in the different solids into a single solid.**
Subtract **(cut) - the operation subtracts the volume of one solid from the other solid object.**
Intersection **- the operation keeps only the volume common to both solids**

Constructive Solid Geometry, CSG



Constructive Solid Geometry



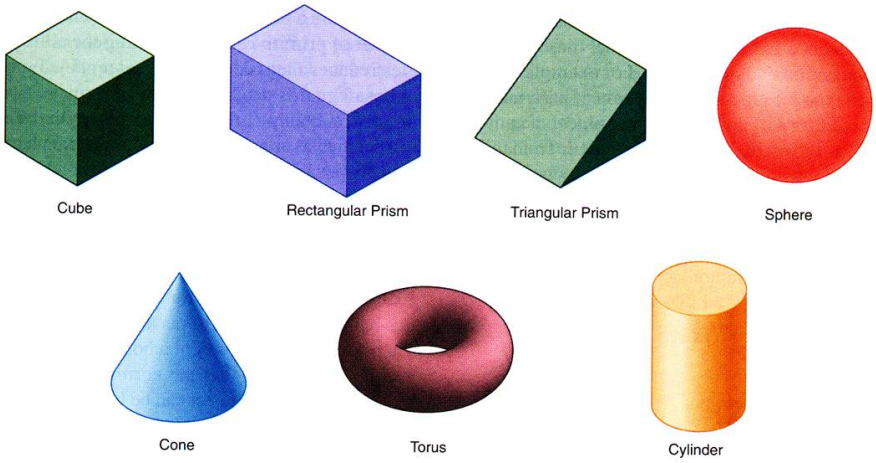
ARISTIDES A. G. REQUICHA

A.A.G. Requicha and R.B. Tilove, Mathematical foundation of constructive solid geometry: General topology of closed rectangular sets. Technical report. TM-27a, PAP, June 1978.

A.A.G. Requicha and H.B. Voelcker,Constructive solid geometry. Technical report. TM-25, PAP, November 1977.

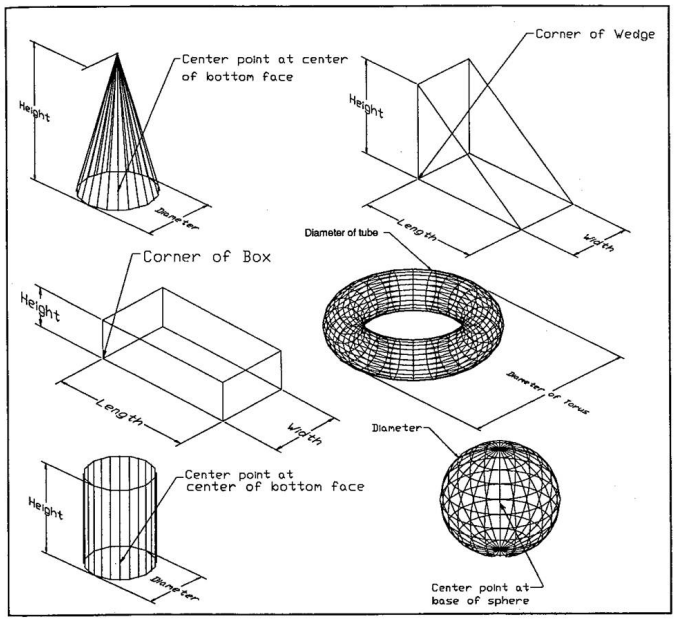
J. Woodwark, Generating wireframes from set-theoretic sold models by spatial division. Computer-Aided Design Vol. 18, No. 6, 1986, 307-315.

Basic Primitive Solids



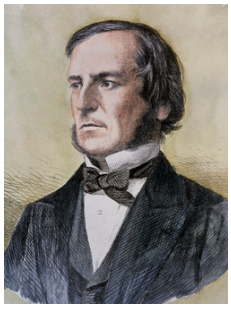
Primitive Solids

The location of the insertion base or base point and default axes orientation.



Boolean Algebra

George Boole, 1815-1864, was an English mathematician and invented Boolean algebra.



Union Intersection Difference

Boolean Operations - Union

union (∪) : Operation places all of the areas enclosed by the original primitives into a new primitive.

the sum of all points in each of two defined sets (logical “or”)

also referred to as *Add, Combine, Join, or Merge*

Boolean Operations - Difference

difference (—) : Operation retains the area that was common to the original primitives.

the points in a source set minus the points common to a second set.

(logical “not”)

sets must share common volume

also referred to as *Subtraction, Remove, or Cut*

Boolean Operations - Intersection

intersection (∩): Operation removes the common area from the first-named primitive.

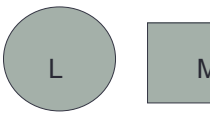
those points *common to each* of two defined sets

(logical “and”)

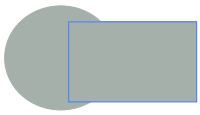
sets must share common volume

also referred to as *Common, or Conjoin*

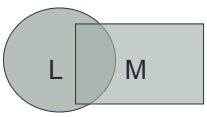
2D Boolean Operations



L ∪ M 2-D union



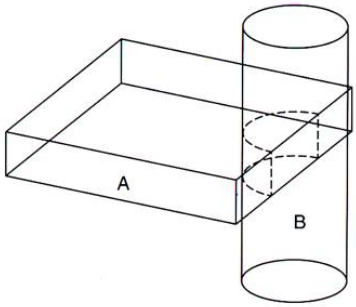
L ∩ M 2-D intersection

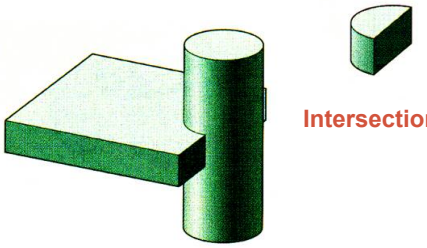


L — M 2-D substraction

M — L 2-D substraction

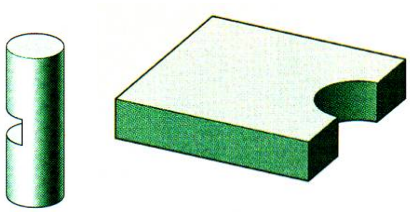
Boolean Operations





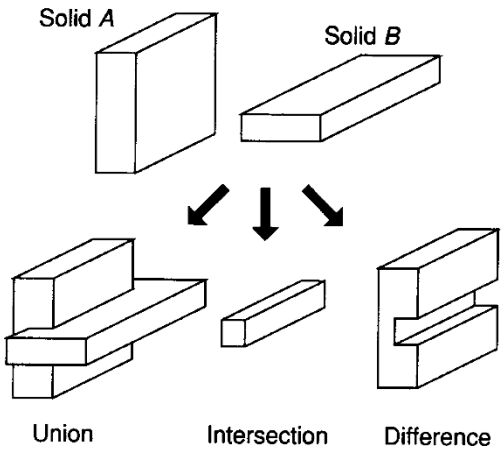
Union

Intersection



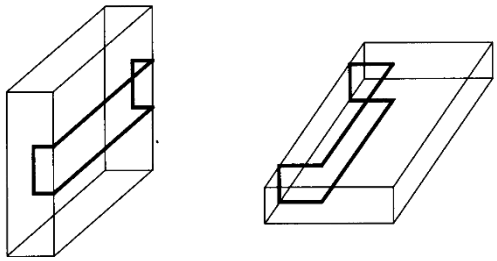
Subtract

Implementing Boolean Operation



Boolean Operation

The intersection curves of all the faces of solid *A* and *B* are calculated. These intersections are inscribed on the associated faces of the two solids.



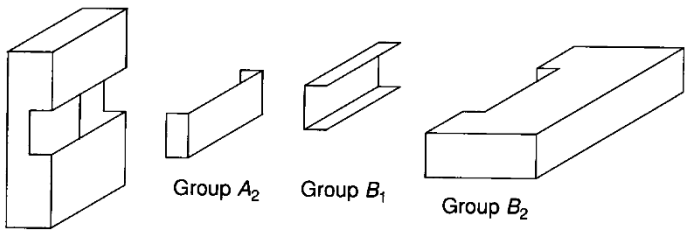
Solid *A* after modification

Solid *B* after modification

Boolean Operation

The faces of solid *A* are classified according to their relative location with respect to solid *B*. Each face is tested to determine whether it is located inside, outside, or on the boundary surface of solid B.

The faces in group A_1 are outside solid *B*, and those of group B_1 are inside solid *A*.



Group A_1

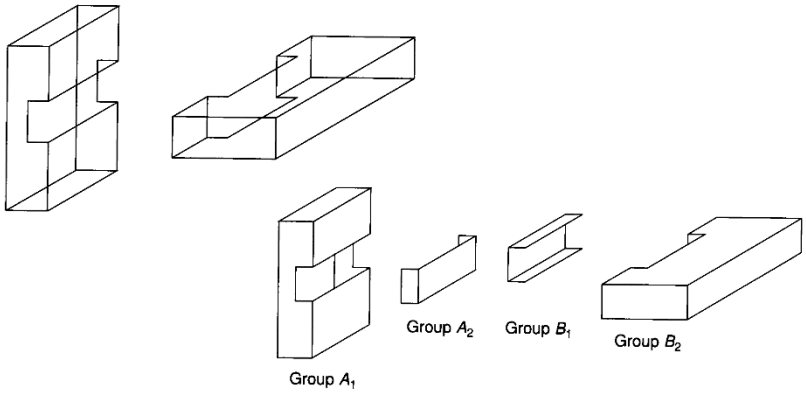
Group A_2

Group B_1

Group B_2

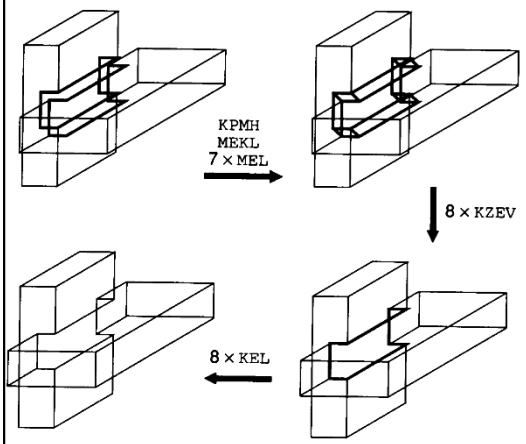
Boolean Operation

Groups of faces are collected according to the specific Boolean operation and the unnecessary face groups are eliminated. For example, for union operation, group A_1 and B_2 are collected and A_2 and B_1 are eliminated.



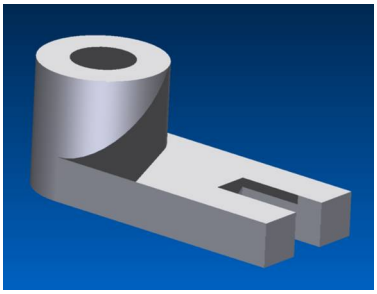
Boolean Operation

The two solids are glued at their common boundary.

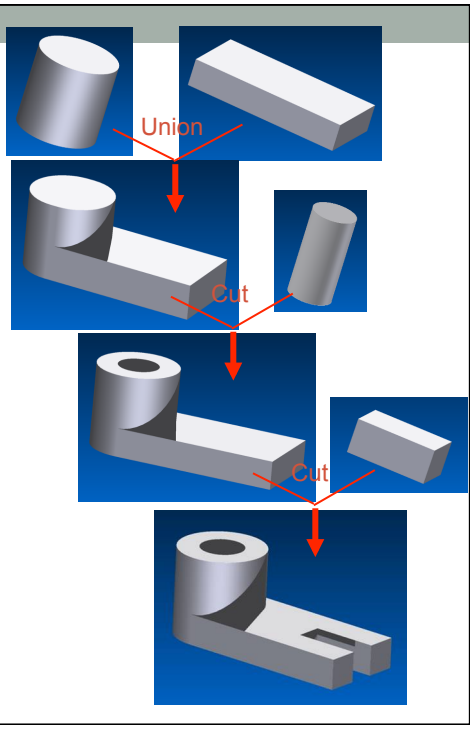


MEVVLS (KEVVLS)	make (kill) edge, two vertices, loop, shell	⇔
MEL (KEL)	make (kill) edge, loop	⇔
MEV (KEV)	make (kill) edge, vertex	⇔
MVE (KVE)	make (kill) vertex, edge	⇔
MEKH (KEMH)	make (kill) edge, kill (make) hole	⇔
MZEV (KZEV)	make (kill) zero length edge, vertex	⇔
MPKH (KPMH)	make(kill) peripheral loop, kill (make) hole loop	⇔ Peripheral loop

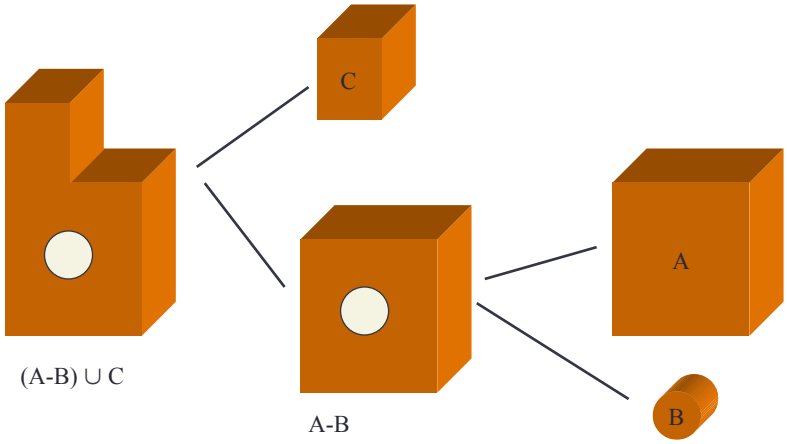
Solid Modeling Example Using CSG



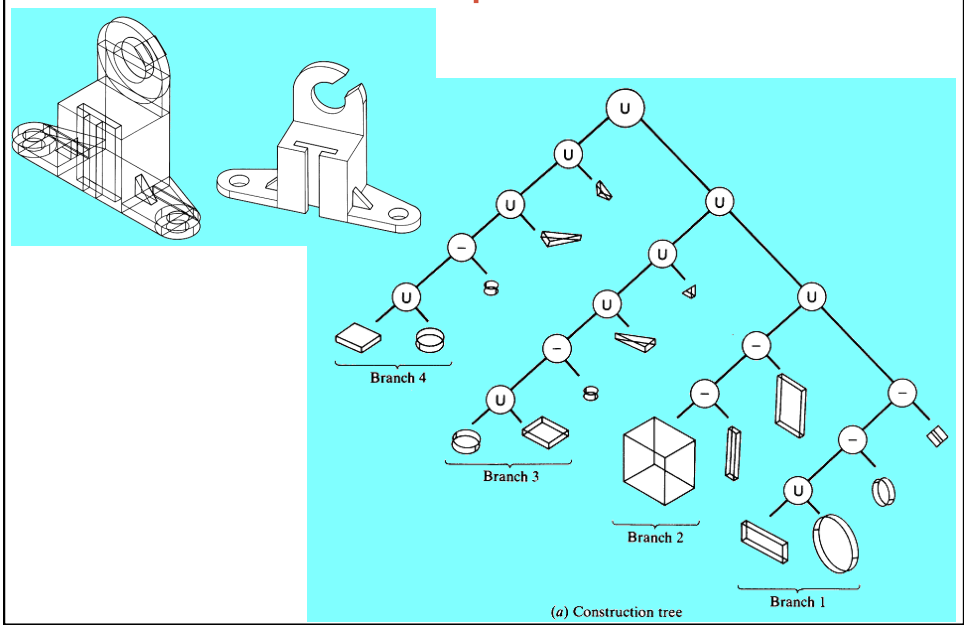
Plan your modeling strategy before you



CSG Example -1



CSG Example -2



CSG vs. B-Rep

CSG	B-Rep
✧Simple representation ✧Limited to simple objects ✧Stored as binary tree ✧Difficult to calculate ✧Rarely used anymore	✧Flexible and powerful representation ✧Stored explicitly ✧Can be generated from CSG representation ✧Used in current CAD systems

4

Sweeping

Sweep representation

The following two categories of solids are considered as 2 ½ dimensional objects:

- Solid of uniform thickness – **Extruded solid**
- Axisymmetric solids – **Solids of revolution**

The first kind is called **Linear Sweep** and the second kind is called **Rotational Sweep**.

Sweeping is used to create objects for B-rep and CSG representations. Thus it becomes an input option in many types of representations.

There is no modeler entirely based on sweeping for the following reasons:

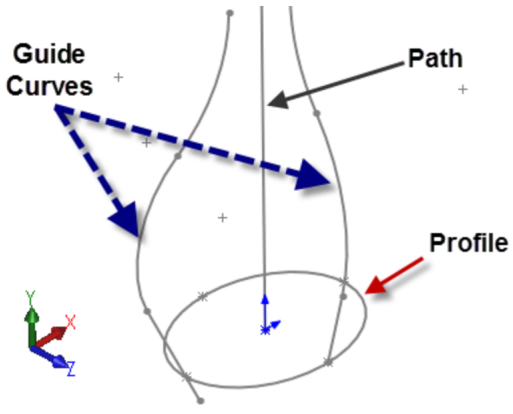
- Limited domain
- No formal theory
- Validation and regularisation schemes are unkown.

Sweeping

- Sweeping can be simple or complex.
- The sweep section dos not vary along the length of the path.
- Sweeping can be much more complex.
- Swept features can be also incorporate in 3-D curves or model edges as paths, and the vary as it moves along a set of other curves called guide curves.

Sweep components

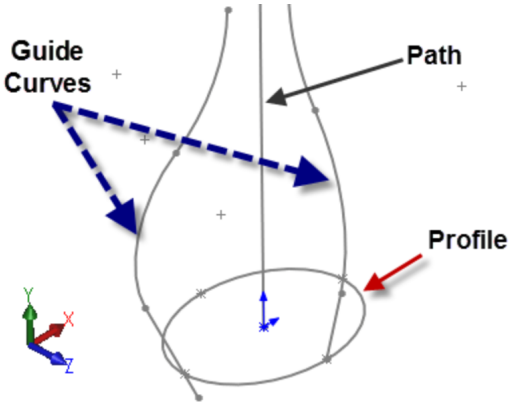
Profile: Sweeping only supports a single profile sketch. It must be a closed, non-self-intersecting boundary. However the sketch can contain multiple contours-either nested or disjoint.



Sweep components

Guide curves: Sweeps can contain multiple guide curves which are used to shape the solid. As the profile is swept, the guide curves control its shape. One way to think of guide curves is to visualize them driving a parameter such as radius.

In this illustration, the profile is attached to the guide curve.
As the profile is swept along the path, the radius of the circle changes, following shape of the guide.



Sweep components

Sweep Path: The helps determine the length of sweep by its endpoints. This means that if the path is shorter than the guides, the sweep will terminate at the end of the path. The system also uses the path to position the intermediate sections along the sweep. Assuming the profile plane is normal to the path:

- The orientation/twist type option Follow path means that the intermediate sections will always stay normal to the path.
- If the Keep Normal constant option is used, the intermediate sections will stay parallel to the plane of the profile sketch.

Sweeping

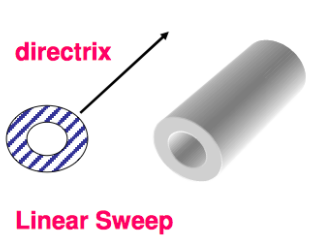
Sweeping and Lofting: What's the difference?

There are also some general differences between sweeping and lofting that will influence which method to use. In essence:

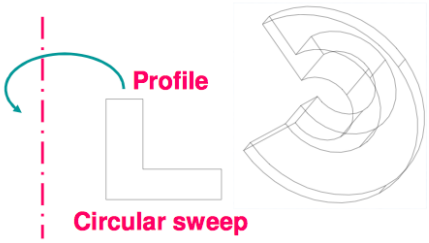
1) Sweeping uses a single profile sketch
2) Lofting uses multiple profile sketch

Types of sweep representation

Closed Profile	Path
1. Linear	Linear or circular
2. Nonlinear	Higher order curves
3. Hybrid	(1)(OP)(2) set operations

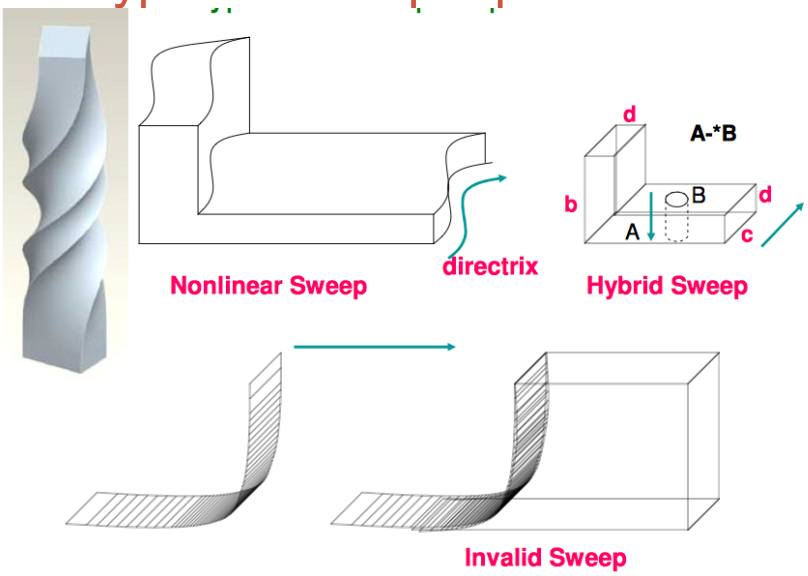


Linear Sweep



Circular sweep

Types of sweep representation



Sweeping

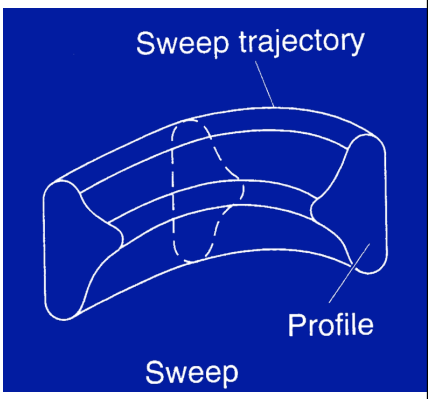
- Most solid geometry is created via sweeping and lofting operations
- Sweeps and lofts are intuitive processes by which to construct objects.
- Sweep and loft model definitions
  - are constructed internally in the system through use of Euler operators.
  - Are constructed externally in system through the use of parent geometry

Sweeping processes

The sweeps can be compared to the extrude option; whereas the extrude option creates a feature by protruding a section along a straight trajectory;

the sweep option creates a section along a user-defined trajectory.

This trajectory can be either user-sketched or selected on the work screen.



Sweeping processes

Extrusions (translational sweeps)

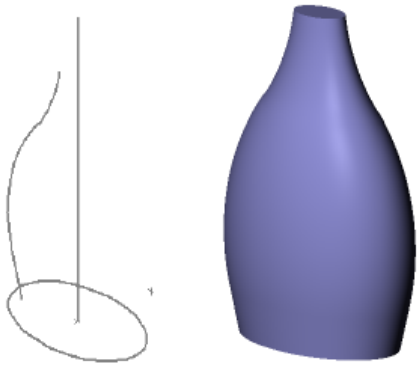
– Volume defined by sweeping specified parent geometry through a direction vector.

– Some software permit scaling of the crosssectional parent geometry may be applied during the sweep to produce a tapering of the object (draft)

Sweeping processes

Solids of Revolution (rotational sweeps)

– Volume defined by the rotational sweep of specified parent geometry through a defined angle, about a defined axis.

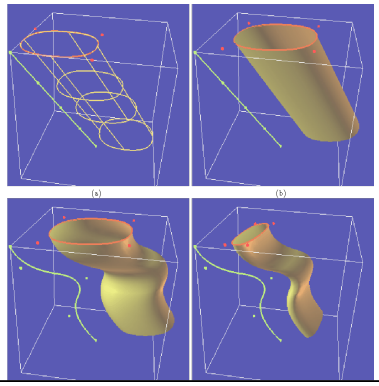


Sweeping processes

General Sweeps

– volume defined by sweeping parent geometry of a section profile (closed) through a path defined by a curve of arbitrary shape

A solid is defined in this scheme in terms of volumes swept out by two-dimensional or three-dimensional as they move along a curve.



Sweeps: Requirements

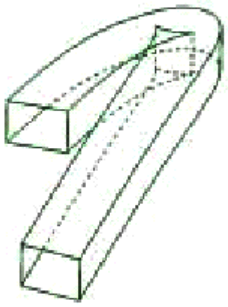
- The sweep profile may consist of multiple segments, but must be continuous (no gaps).
- The sweep profile must form a closed loop in order to create solid geometry.
- The sweep profile must existing within a single plane.
- The sweep path must continuous.

Sweeps: Concerns

- Concerns care must be used in defining sweep geometry such that it does not self-intersect.
- If the sweep path includes arcs or splines of a small radius as compared to the profile being sweep, self-intersection may occur.
- In general keep sweep profiles as simple as possible

Self-Intersecting Sweep

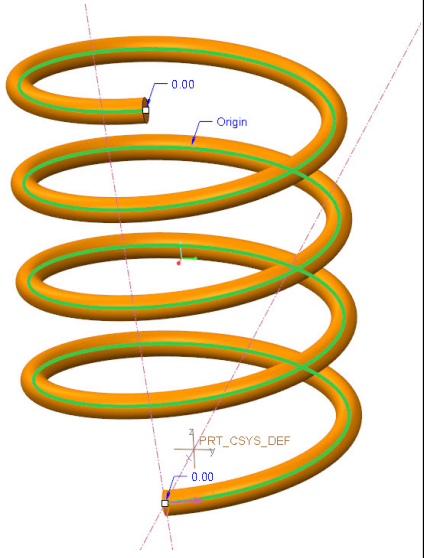
When viewed in wireframe display, the self-intersection is very apparent.



Helical Sweeps

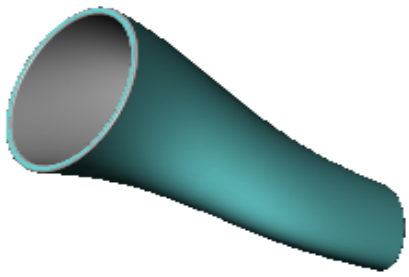
As its name implies, the Helical sweep option is useful for creating parts that consists of helical features.

Two features often created with the helical sweep options are spring and threads.



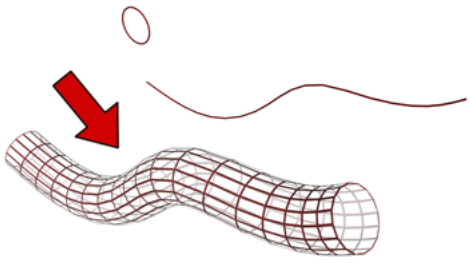
Lofting

Lofting is typically used to create objects of varying cross section; however, lofting could also be used to create objects of constant cross section in which the cross section is copied and oriented in space along something other than a straight-line path.



Lofting

- Defined by 2 or more cross-section curves (profiles or section curves)
- Spline curves fitted to corresponding points on each profile to define the boundaries of the loft's faces
- Lofts
- Spline curves constructed from profiles
- Final lofted
- Geometry



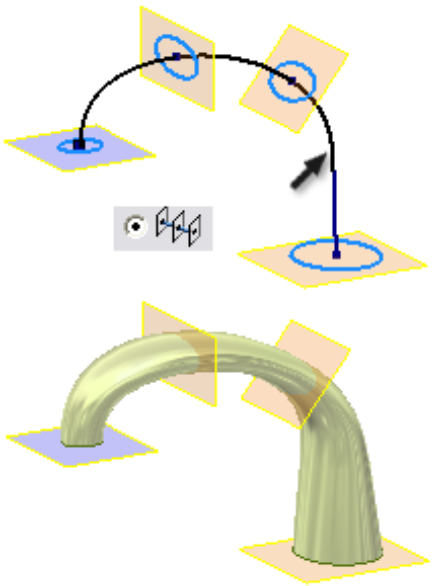
Lofting

Orientation of profiles may be limited

– may restrict profiles to parallel planes

Number of curve segments per profile may be limited

– may require the number of curve segments per profile to be the same

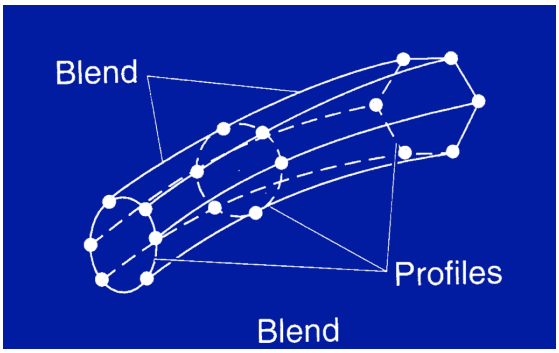


Blended Features

The blend option can also be compared to the extrude option. Primarily, the blend option creates a feature by protruding along a straight trajectory between two or more user-defined sections. A partially revolved blend can be created also.

Three types of blends are available:

- parallel
- rotational
- general

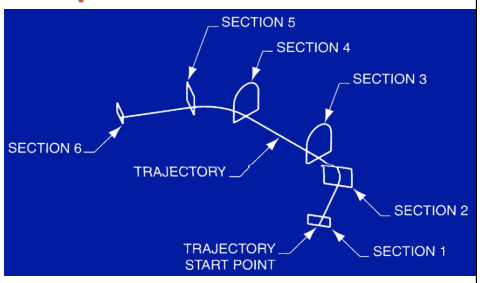


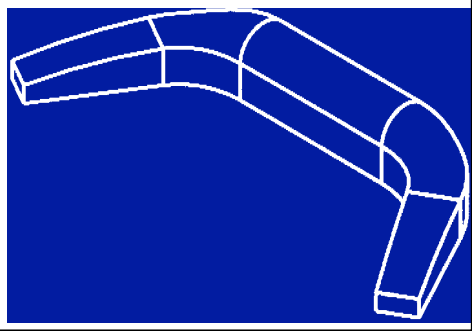
Swept Blend Option

A swept blend is a combination of a sweep and a blend. A swept feature is a section protruded along a defined trajectory.

This trajectory can be either sketched or selected.

A parallel blended feature is a feature protruded along a straight trajectory between two or more user-defined sections.





5

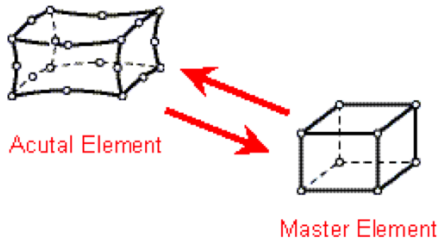
Analytical Solid Modeling

Analytical Solid Modeling

Historically it is closely related to 3-D isoparametric formulation of FEM for 8- to 20-node hexahedral elements.

This arose from the need to model complex objects for FEA.

ASM uses the parametric representation of an object in 3-D space that is a mapping of a cubical parametric domain into a solid described by the global coordinates.



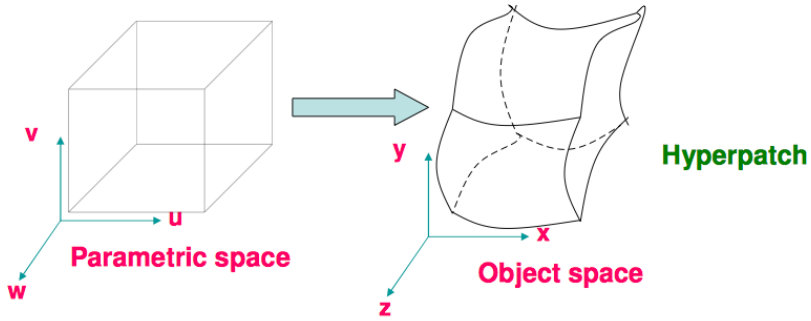
Analytical Solid Modeling

- ASM is mainly for design applications and not suited for manufacturing.
- It does not involve orientable surfaces as in B-rep and CSG.
- Applications :
    1. Mass property calculation
    2. Composite material modeling
    3. Computer animation
    4. FEM mesh generation with hyperpatch concepts.

E.g. PATRAN-G is based on ASM and has interface to various FEA packages.

Analytical Solid Modeling

ASM is an extension of bi-parametric surface representation to 3-D parametric space. Tensor product method is used as used in the case of surface.
ASM solids as tricubic, Bezier and B-spline solids analogues to bicubic, Bezier and B-spline surfaces in 2-D parametric space (u,v) and analogous to corresponding types of curves in one-dimension (t)



Analytical Solid Modeling

Hyperpatch : The parametric solid is called hyperpatch as it is extension of and bounded by surface patches. The points in the interior and on the boundary of the parametric solid is given by

$$P(u,v,w) = [x \quad y \quad z] = [x(u,v,w) \quad y(u,v,w) \quad z(u,v,w)]$$

$$u_{\min} \le u \le u_{\max}; v_{\min} \le v \le v_{\max}; w_{\min} \le w \le w_{\max}$$

Building operation : The object is divided into hyperpatches. The hyperpatches are represented using surfaces and curves. A cubic polynomial in each parameter is sufficient for most practical applications

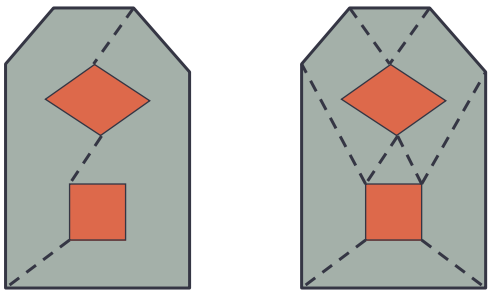
$$P(u,v,w) = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} C_{ijk} u^{i-1} v^{j-1} w^{k-1} \qquad 0 \le u \le 1; 0 \le v \le 1; 0 \le w \le 1$$

6

Cellular Decomposition

Cellular decomposition

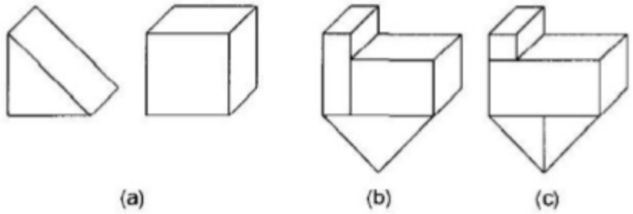
An object is represented in this scheme by a list of the cells it occupies, but the cells are not necessarily cubes, nor are they necessarily identical.



Cellular decomposition

- Defines a set of primitive cells
- Differs from primitive instancing
  (Glue operation)
- Unambiguous but not necessarily unique.

Difficult to validate



Cellular decomposition

Two classes of methods:

- Exact cell decomposition
- Approximate cell decomposition
  F is represented by a collection of non-overlapping cells whose union is contained in F
  Examples: quadtree, octree, 2^n-tree

Exact Cell Decomposition

Step 1: is to decompose the free space, which is bounded both externally and internally by polygons, into trapezoidal and triangular cells by simply drawing parallel line segments from each vertex of each interior polygon in the configuration space to the exterior boundary.
Step 2: each cell is numbered and represented as a node in the connectivity graph.
Step 3 : Nodes that are adjacent in the configuration space are linked in the connectivity graph.
Step 4 : A path in this graph corresponds to a channel in free space, which is illustrated by the sequence of striped cells.
Step 5: This channel is then translated into a free path by connecting the initial configuration to the goal configuration through the midpoints of the intersections of the adjacent cells in the channel.

Approximate Cell Decomposition

This approach to cell decomposition is different because it uses a recursive method to continue subdividing the cells until one of the following scenarios occurs:

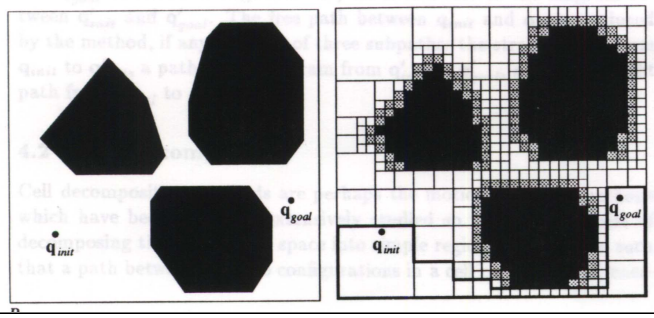
- Each cell lies either completely in free space or completely in the C-obstacle region

- An arbitrary limit resolution is reached.

Approximate Cell Decomposition

Once a cell fulfills one of these criteria, it stops decomposing. This method is also called a "**quadtree**" decomposition because a cell is divided into four smaller cells of the same shape each time it gets decomposed.

After the decomposition step, the free path can then easily be found by following the adjacent, decomposed cells through free space.



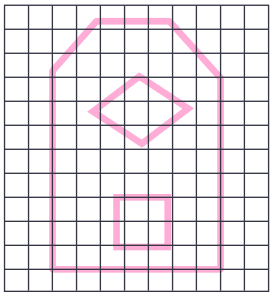
7

Spatial occupancy enumeration

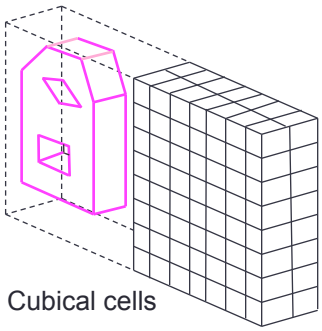
Spatial occupancy enumeration

For this technique three-dimensional space is divided up into cubical cells at a particular resolution and objects are modelled by listing the cells that they occupy.

Object and 2D grid of cells



Object

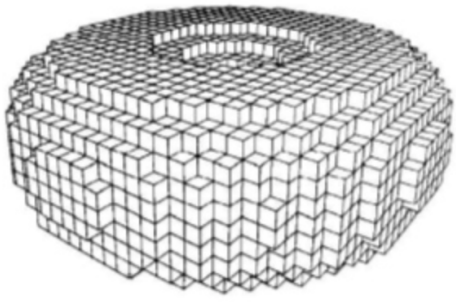


Cubical cells

Spatial occupancy enumeration

In this scheme solids are decomposed into cells each with a simple topological structure and often with a simple geometric structure.

For example, define solid point set in terms of collection of non-overlapping "blocks".



Spatial occupancy enumeration

Advantages

- Often used in biomedical application
- Unique and unambiguous
- Approximate
- Save spaces

8

Octree Encoding

Octree Encoding

The Octree Encoding scheme is similar to both the spatial enumeration and cell decomposition approaches noted previously.

The information contained in the encoded representation of an object is identical to that available in the spatial enumeration representation.

Octree Encoding

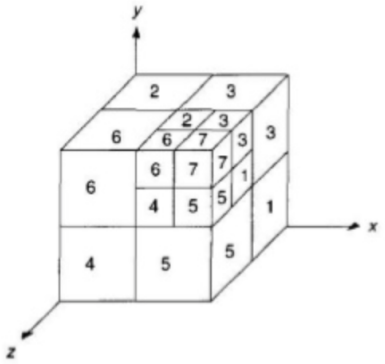
Octree Encoding has been developed to realize the goals :

(1) To develop a capability to represent any 3-D or N-dimensional object to any specified resolution in a common encoding format
(2) To operate on any object or set of objects with the Boolean operations and geometric operations
(3) To implement a computationally efficient (linear) solution to the N-dimensional interference problem.
(4) To develop the capability to display in linear time any number of objects from any viewpoint with color, shading, shadowing, multiple illumination sources, transparent objects, orthographic or perspective view and smooth edges
(5) To develop a scheme that can be implemented across large numbers of inexpensive high-bandwith processors that do not require floating-point operations, integer multiplication or integer division.

Octree Encoding

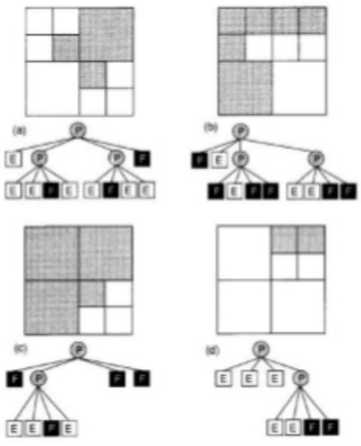
The "Octree" is similar to the quadtree

(LUF,LUB,LDF,LDB,RUF,RUB,RDF,RDB)



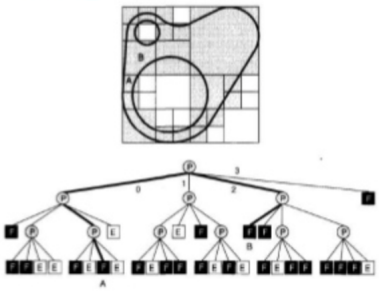
Octree Encoding

Boolean set operations and transformations



Octree Encoding

Neighbor finding

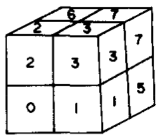


Linear notations

– 00X,010,011,020,.....

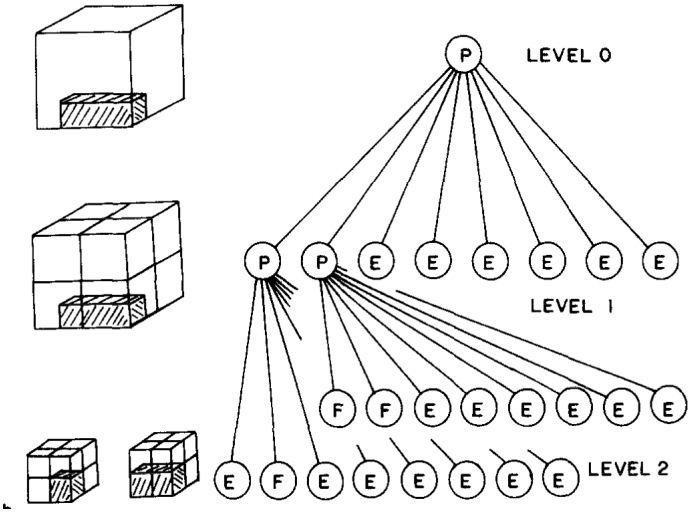
Simple object represented in Octree Encoding format

Numbering sequence and label definitions.

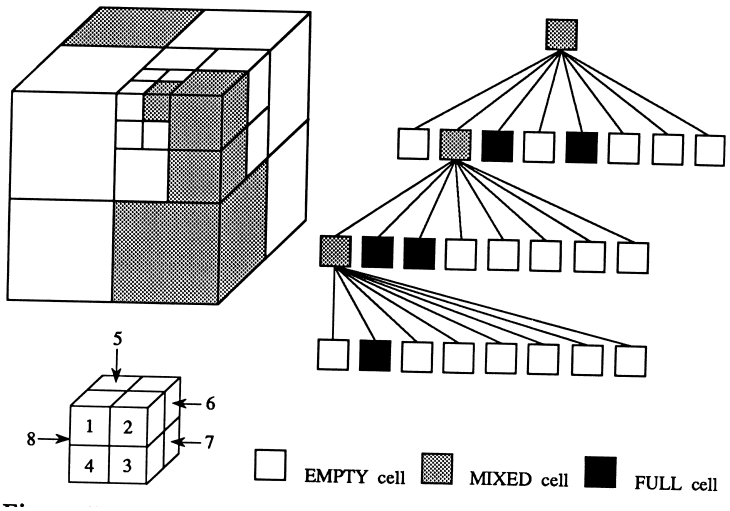


EMPTY ("E")- CUBE IS EMPTY
FULL ("F")- CUBE IS COMPLETELY ENCLOSED BY THE OBJECT
PARTIAL ("P")- CUBE PARTLY INTERSECTS OBJECT

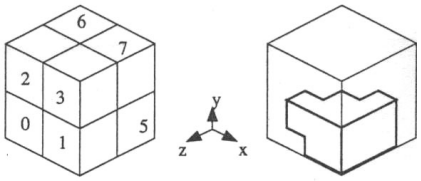
Three-level tree representation of an object



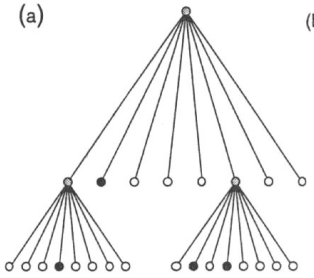
Octree decomposition



Octree decomposition



(a) (b)



(c)

Octree data structure

```
struct octreeroot
{
   float xmin, ymin, zmin;    /* space of interest */
   float xmax, ymax, zmax;
   struct octree  *root;/* root of the tree */
};

struct octree
{
   char  code;                /* BLACK, WHITE, GRAY */
   struct octree  *oct[8];    /* pointers to octants, present if GRAY */
};
```

9

Primitive Instancing

Pure primitive instancing

Point set represented as a combination of more basic primitive point sets

– each primitive point set is an instance of a defined primitive solid
– first efforts were called primitive instancing

Pure primitive instancing

- Define a set of primitive 3-D solid shapes
- Primitive instances are similar to parameterized objects
- A family with few difference in members
- Relatively complex objects
- Without combining objects

Pure primitive instancing

Every scheme has primitives, which must be instantiated to construct structured representations.

Primitives may be low level, e.g. the points used to represent polygons in the previous section. But they also may be high level.

For example, many modeling system have solid primitives such as blocks and cylinders.

Pure primitive instancing

A primitive is a parameterized geometric entity, typically represented by a tuple containing a type code (e.g. a string 'point' or 'block') followed by several real numbers that correspond to specific parameter values.

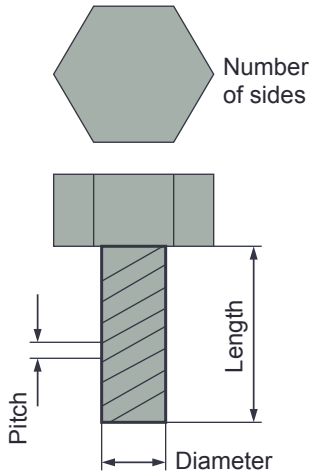
For example, a solid block aligned with the principal axes can be represented by the 4-tuple

('block', XSize, YSize, Zsize)

Pure primitive instancing

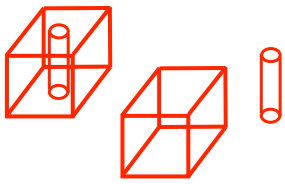
Modellers using this technique can handle a number of families of objects, each defined parametrically.

BOLT (NSIDES,LENGTH, PITCH, DIAMETER)



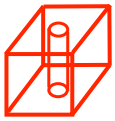
Approaches

Constructive Solid Geometry (CSG)



Solids CSG:
1 Block primitive
1 Cylinder primitive
1 Equation : Block-Cylinder

Boundary Representation (B-rep)



Solids Brep:
7 Faces
8 Vertices
14 Edges

Sweeping

Solids Swept:
1 Base block
1 Circular profile
1 Straight profile to sweep the cutting circle through the block

Approaches

Hybrid
(feature-based modelers)

Features:
1 Block from stock
1 Bored hole

Analytical Solid Modeling
(ASM) *P=P(u,v,w)*

Block Hole:

```
HOLE = { radius = 5
         height = 5
         x_position = 1.5
                  y_position = 1.5
                  x_rotation = 90
                  radius_tolerance = 0.001
         }
```

SKETCHING & FEATURES

Sketching and Features

When discussing the mind-set needed for working with parametric modelers, there are two topics that need to be expanded: ***Sketching and Features***

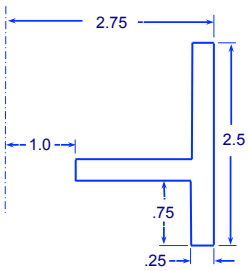
Sketching

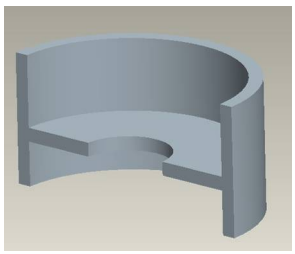
- Take the word sketch literally. A sketch should be just that, a *sketch*.
- When sketching, it is not necessary to create geometry with accuracy. Lines, arcs, and additional geometry need not be created with exact dimensions in mind.
- When the dimensions are added, the sketch will change size and shape. This is the essence of Parametric Modeling.

Sketching and Features

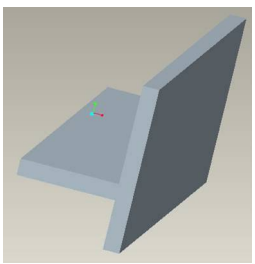
Features

- *Sketched Feature*
  - ➢ Create a 2D sketch.
  - ➢ Create a feature from the sketch by extruding, revolving, sweeping, lofting and blending.



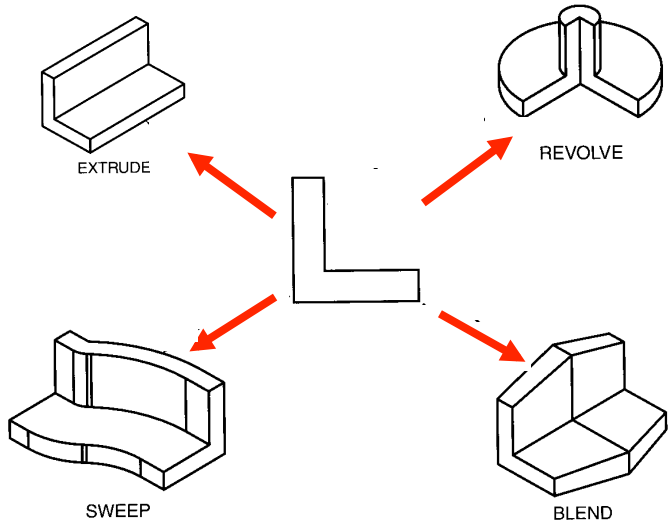


Revolved feature



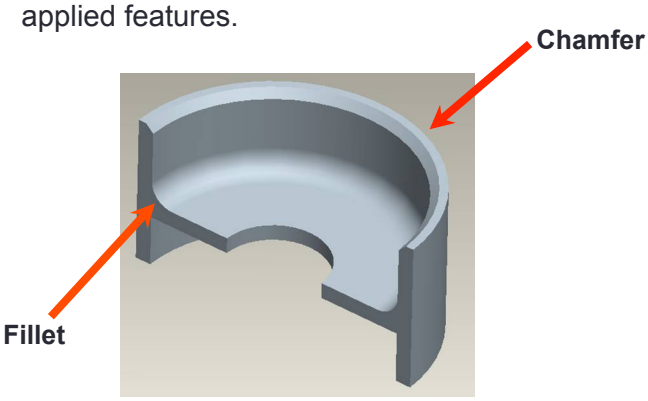
Extruded feature

Creating Features from Sketches



Sketching and Features

- *Applied Feature*
  - Applied feature does not require a sketch.
  - They are applied directly to the model.
  - Fillets and chamfers are very common applied features.



Summary – Solid Modeling Methods

- **Primitive creation modeling**

  A solid model is created by retrieving primitive solids and performing Boolean operations
- **Sweeping function**

  Creates a solid by translating, revolving or sweeping a predefined 2D shape ***(Sketching).***

  If geometric and dimensional constraints are imposed, it is called ***Parametric Modeling***.
- **Feature-based Modeling**

  Models a solid by using familiar shapes; holes, slots, grooves, pockets, chamfers, fillets…..

Summary – Solid Modeling Methods

- **Modifying functions**

  Rounding (or blending) and lifting.

- **Boundary Modeling**

  Lower entities of a solid (vertices, edges and faces) are directly manipulated.

Sketching

- **Take the word sketch literally. A sketch should be just that, a *sketch*.**
- Sketches are able to capture the designer's intent for the part like no other technique.
- The sketch is the best tool for creating solids because of flexibility in shapes and the ability to edit.
- Ideal for creating unusual freeform shapes.

Sketching

- A sketch is a set of two dimensional curves joined in a string that when swept or extruded forms a solid body.
- When sketching it is not necessary to create geometry with accuracy.
- Lines, arcs, and additional geometry need not be created with exact dimensions in mind.

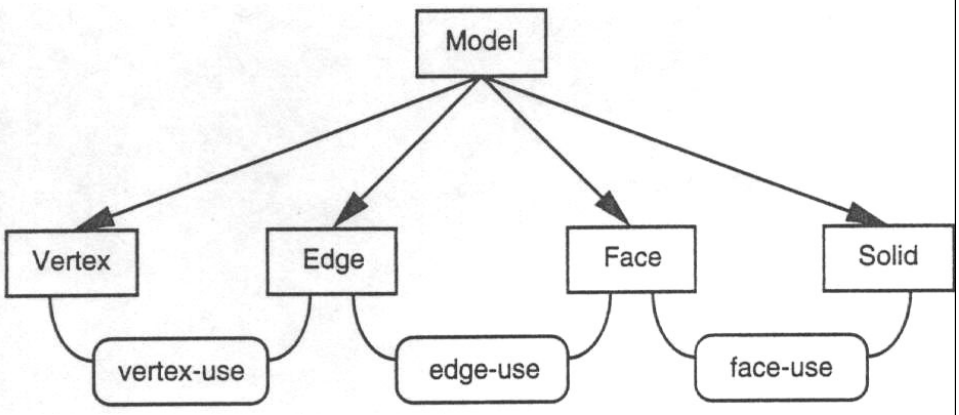
Sketching

- When the dimensions are added, the sketch will change size and shape. This is the essence of ***parametric modeling***
- In short, the sketch need only be the approximate size and shape of the part being designed. The geometric constraints and dimensions, when added, will drive the size and the shape of the geometry.
- Curves are parametrically associated to each other and the solid that is created by them.

2 types of non-manifold modelers

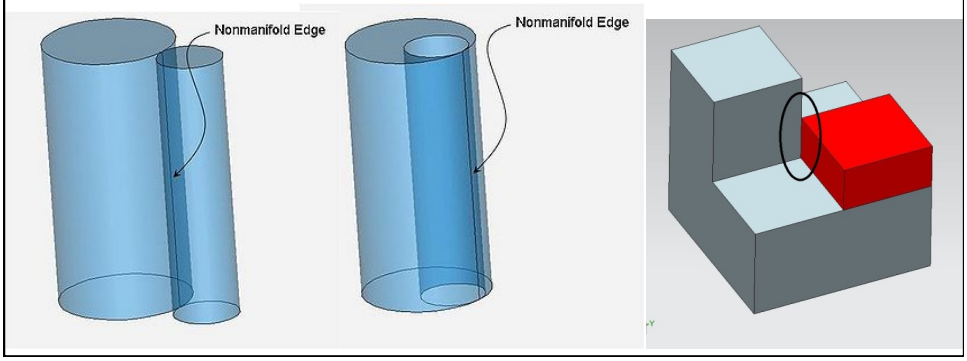
- Include lower dimensional entities in a solid model data structure
- Concentrate on composite models => cellular model

Non-manifold data structure



Manifold vs.Non-manifold geometry

- Mathematically manifold geometry means every point on a surface has “neighborhood” (infinitesimal sphere) around it that can be deformed onto a locally planar surface.
- More simply: manifold geometry rules out objects which are not physically realizable such as those with features joined along a single edge or vertex.



Manifold vs.Non-manifold geometry

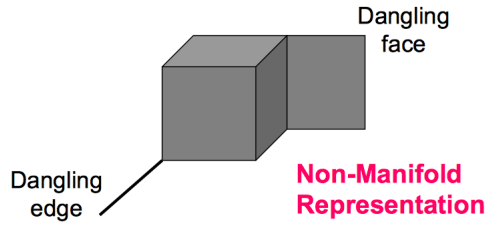
Two Manifold Representations:

Two manifold objects are well bound, closed and homomorphic to a topological ball.

Non-manifold Representations:

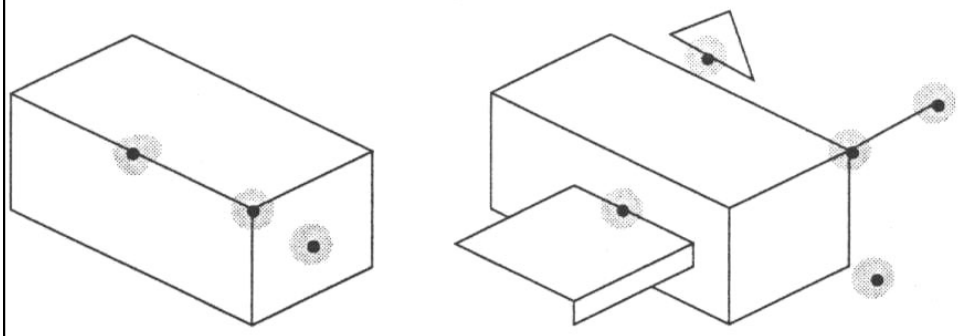
When restrictions of closure and completeness are removed from the solid definition, wireframe entities can coexist with volume based solids.

Two-Manifold Object



Non-Manifold Representation

Non-manifold neighborhood configuration

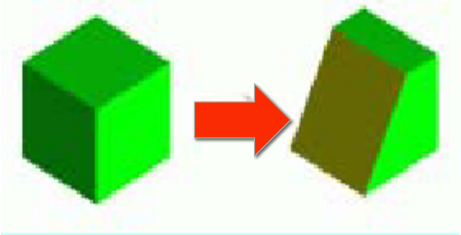


Editing a Solid Model

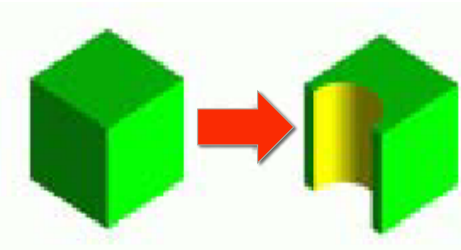
- Use the reordering of object creation sequence to avert failures
- Avoid offsetting surfaces/solids to fix intersection errors
- Editing which alters only geometry* maintains validity *Moving edges, faces
- Editing which changes topology* may violate validity checks, causing errors

*Any edit which changes the number of vertices and/or edges in a face

Editing a Solid Model



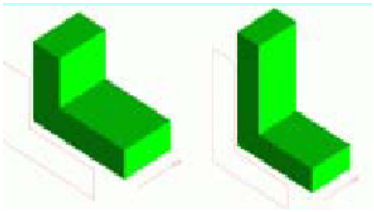
Example of a geometric edit
(no change in topology)



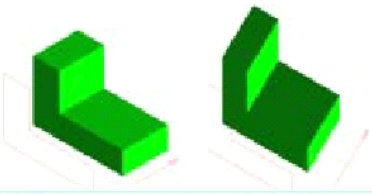
Example of a topological edit
(change in the number of edges, and faces)

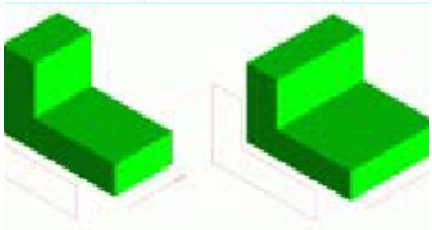
Editing an Extrusion Feature

Changing the Extrusion Profile



Change the Extrusion Direction



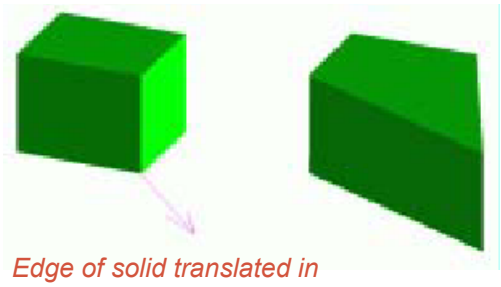


Change the Extrusion Distance

Editing by Tweaking Operations

Other operations, which do not affect the number of vertices, edges, and loops, and hence need not obey Euler's Law may be included.

These allow the geometry to be "tweaked" by moving theses vertices, edges and loops "small" distances.



Edge of solid translated in direction of arrow

CONSTRAINED BASED MODELING

Constraint-based Modeling

Constraint-based modeling is a technique that can help the CAD operator manage the model modification process.

In a constraint-based modeler, describing the relationship of geometric elements with equations and logical relationships creates a part.

Constraint-based systems

Concept first developed in 1970's

First commercial system in 1988

Most current professional level feature-based modelers use constraint-based techniques to help define features.

(through the creation of profiles)

Constraints-based Modeling

There are two major types of constraint-based modeling systems available on the market today: variational geometry and parametric modeling.

A *variational geometry* is a set of plane curves and lines to which you can assign *geometrical and dimensional constraints* and which you can use to create solids or to add features to existing solids.

Variational Geometry

With variational geometry, constraints are applied to the 2-D shape in terms of degrees of freedom.

A degree of freedom exists if there is currently no constraint applied to control a specific feature of the geometry of your sketch.

Types of Constraints

- Ground constraints
- Dimensional constraints
- Geometric constraints

Ground Constraints

Ground constraints are added to your sketch to define which of the pair of lines involved is the movable line.

Ground constraints
- Vertical
- Horizontal
- Angular
- X,Y

Dimensional Constraint

Dimensional constraints specify the length, radius, or rotation angle of the geometric ele-ments in your sketch. Geometric constraints force the profile to conform to a specific shape.

Dimensional constraints, on the other hand, add the parametric characteristic to the profile. You can change a dimension at any time and its new value will be immediately reflected in the design.

Geometric Constraint

Geometric constraints control the shape of the profile.

Geometric constraints are used to define the shape of your 2-D geometry.

Gemetric constraints
- Project
- Join
- Xvalue
- Yvalue
- Radius
- Parallel
- Perpendicular
- Collinear
- Coincident
- Tangent
- Horizontal
- Vertical

PARAMETRIC MODELING

Parametric Modeling Concept

- ***Parametric*** is a term used to describe a dimension's ability to change the shape of model geometry if the dimension value is modified.
- ***Feature-based*** is a term used to describe the various components of a model. For example, a part can consists of various types of features such as holes, grooves, fillets, and chamfers.
- Parametric modeler are featured-based, parametric, solid modeling design program: **SolidWorks, Pro-Engineer, Unigraphics (CSG and parametric),** …..

Parametric Modelling

a subset of variational geometry.

allows for a flexible design in much the same way that variational modeling does.

able to add constraints to sketch to control how the geometry will behave as modifications are made.

need to have a fully constrained shape before using parametric modeling to create a 3-D solid.

Parametric & Variational Modelling

Parametric modelling

- constraints defined sequentially
- each constraint calculated based on previously defined constraints
- order of constraint specification is important

Variational modelling

- constraints solved simultaneously
- order of constraint specification doesn't matter

What is a profile?

A *variational profile* is a set of plane curves and lines to which assigning *geometrical and dimensional constraints* and which using to create solids or to add features to existing solids.

Variational profiles are the basic components of parametric structure.

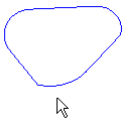
One can use profiles to create solids and features by applying a linear sweep or revolving the profile.

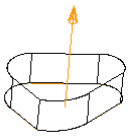
In the case of a feature, the profile is used either to add material to or remove material from a solid.

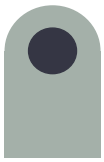
Rules for building a profile

You can create a profile from any set of curves that meet the following conditions:

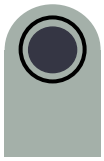
- The curves must be co-planar and be drawn on the Work Plane.
- The curves can form an open or closed profile and may contain islands, but **not** *islands within islands*.







A closed profile containing one island can be created from this set of curves

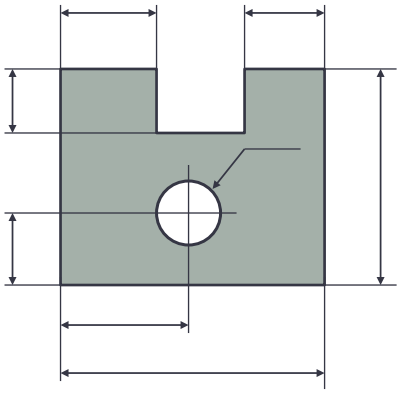


A sweep solid cannot be created from this set of curves

Parametric & Variational Modelling

Parametric definition

User specifies dimension D_1, other dimensions calculated sequentially



$D_1 = x$

$D_2 = D_1/2$

$D_3 = {}_2\,D1$

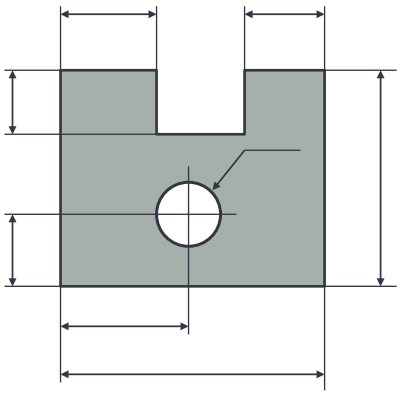
$D_4 = D_3 / 2$

$D_5 = 50$

Parametric & Variational Modelling

Variation definition

Solve system of simultaneous equations:



$$D_1 - 2D_2 = 0$$

$$2D_1 - D_3 = 0$$

$$D_3 - 2\,D_4 = 0$$

$$D_1 - x = 0$$

$$D_5 - 50 = 0$$

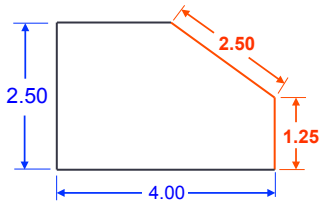
Design Intent

- In parametric modeling, dimensions control the model.
- Design intent is how your model will react when dimension values are changed.

Design Intent

Example:

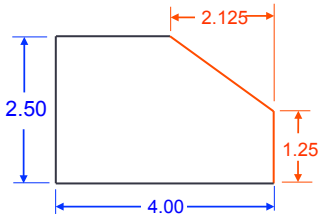
The drawing shows the intent of the designer that the inclined plane (chamfer) should have a flat area measuring 2.5 inches and that it should start at a point 1.25 inches from the base of the drawing. These parameters are what the designer deemed significant for this model.



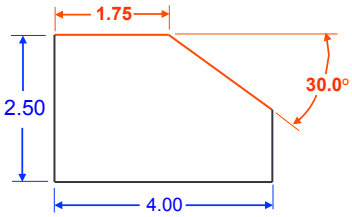
Remember that the placement of dimensions is very important because they are being used to drive the shape of the geometry. If the 2.5 in. vertical dimension increases, the 2.5 in. flat across the chamfer will be maintained, but its angle will change.

Design Intent

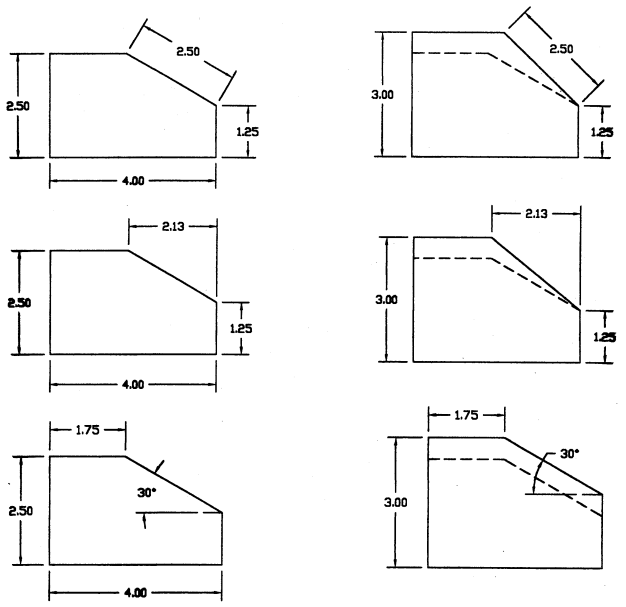
In this drawing, what is important to the designer is the vertical location and horizontal dimension of the chamfer, rather than the flat of the chamfer.



In the last drawing, the designer calls for a specific angle for the chamfer. In this case the angle of the chamfer should be dimensioned.



Design Intent



Boolean vs. Parametric Modeling

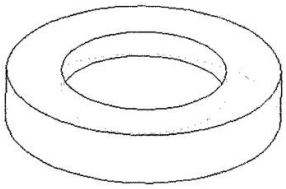
The ability to go back on some earlier stage in the design process and make changes by editing a sketch or changing some dimensions is extremely important to a designer.

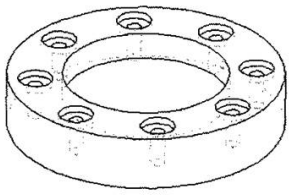
This is the main advantage of a ***parametric*** (SolidWorks, Unigraphics, Inventor, Pro-Engineer) over a ***non-parametric*** modeler (AutoCAD 3D modeler – Boolean operation)

Boolean vs. Parametric Modeling

Example:

Let's assume that it is desired to design a part consisting of a ring with a certain thickness and a series of counterbore holes along the perimeter.

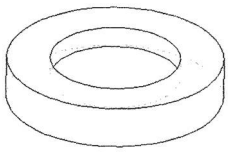




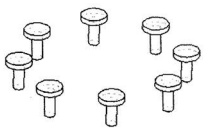
Boolean vs. Parametric Modeling

Boolean operation

Make the base part by creating two cylinders and subtract the small one from the large one

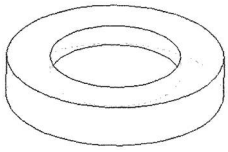


Create the solid geometry that will become the counterbore holes and generate the pattern.

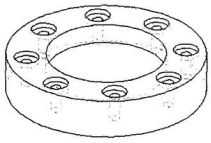


Boolean vs. Parametric Modeling

Position the pattern about the perimeter of the base part. Locating the holes is critical to creating an accurate solid model.



Subtract the pattern from the base part to create the actual holes.



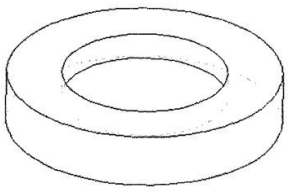
What would happen if you had to come back to this part to change the thickness of the ring or size of the counterbore holes?

Since Boolean operation was used to create the part, changing the thickness would not increase the height of the holes. There is no association between the thickness and the hole pattern location.

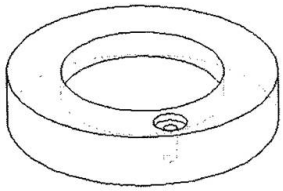
Boolean vs. Parametric Modeling

Parametric modeling (SolidWorks, ProE, UG, …)

Create the initial base, the ring, by extruding the profile (circles) in a particular direction (Pro/E or SolidWorks) or use primitive solids and Boolean operation (UG).

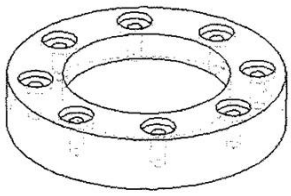


Create the counterbore as a feature. Select the top surface of the ring and either sketch the two holes and extrude at different depth or use the hole feature option.



Boolean vs. Parametric Modeling

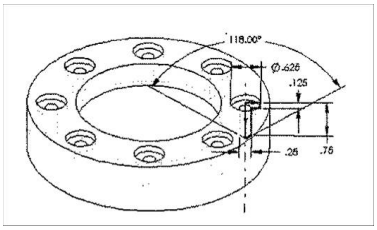
The next step would be to pattern the hole. The pattern would actually be considered a feature in itself, and would have its set of parametric variables, such as the number of copies and the angle between copies.



The model created would be identical to the one created using Boolean operation, **but with intelligence built into the model.**

Boolean vs. Parametric Modeling

The true power of parametric modeling shines through when ***design changes*** need to be made. The design modification is made by simply changing a dimension.



Since the counterbore is associated with the top surface of the ring, any changes in the thickness of the ring would automatically be reflected on the counterbore depth.

FEATURE BASED MODELING

Feature Construction Techniques

Feature-based modeling is an attempt to make modeling a more efficient process more in tune with how designers and engineers actually work.

An important advancement in 3-D solids modeling is the introduction of feature-based modeling.

Feature-based modeling has, among engineers, rapidly become the preferred method of constructing solid models.

Feature ConstructionTechniques

Most modern CAD software employs a methodology know as feature-based modeling.

Software packages :
Mechanical Desktop
Pro/E

Feature-based Modeling

- Geometry is defined in terms of real world "features" as opposed to abstract geometric entities.
- For example:

– work with holes as opposed to cylinders

– cuts and extrusions rather than blocks and wedges.

Feature-based Modeling

- Features capture “design intent”
- Included information defines how the features behaves in editing

– for example, a “through” hole

- Features store non-graphic information for use in:

– 2D drawing creation, FEA, CNC and kinematic analysis

Feature-Based Modelling

Advantages,

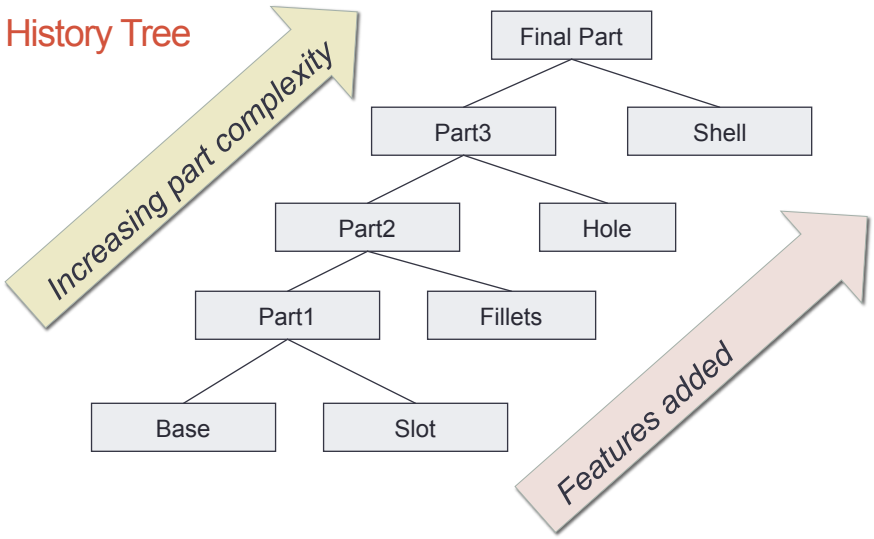
- very intuitive and easy to use
- can simplify other aspects of CIM

  (eg. If a standard feature is used there will be a standard process plan to make that feature).
- emphasizes the use of standard components.

Disadvantages,

- restrictive when dealing with nonstandard features
- interaction of features can be hard to estimate
- a complete set of all possible features would be very large.

Feature-Based Modelling



ASSEMBLY MODELING

Assembly modeling

- Parts modeling and representation
- Hierarchical relation
- Mating conditions
- Dependency among parts
- Hierarchical relation and mating condition
- Mating condition and spatial relation
- Assembly analysis: interference(Mating tolerance), mass property, kinematics /dynamics
- Design change=> auto updates of the assembly

187

Applications of Assembly Models

- Interference checking
- Visualization
  - rendered
  - exploded
- Animation
- Mechanism analysis

Assembly Modelling

Assembly modeling allows the integration of design and manufacturing to production planning and control.

Assembly Model used for ...

- Creation of orthographic assembly drawings.
- Creation of exploded assemblies.
- Facilitate packaging
- Perform interference and clearance checks.

Assembly Modelling

Relationship data includes

– Constraint information

- orientation and location of components with respect to one another
- variational relationship between features of different parts

Assembly Modelling

Relationship data includes ...

– Instancing information

• multiple occurrences of the same component

– Tolerance and fit information

• part interference and clearance

Part/Sub-assembly Placement

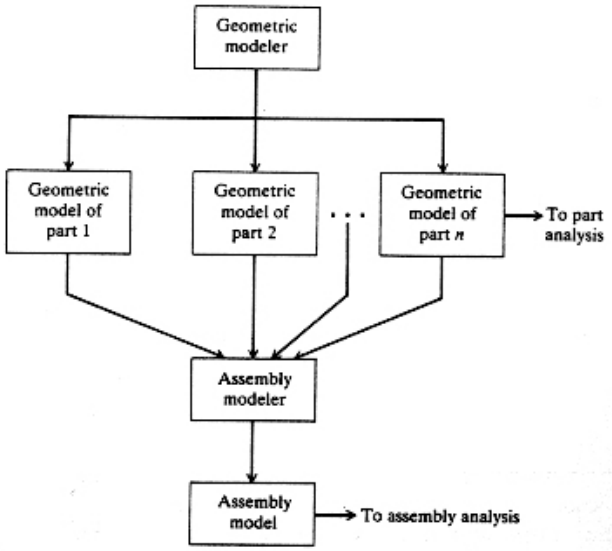
Examples of constraints applied to assemble components

– alignment
  • surfaces, axes
  • with offset distances

– mating
  • surfaces, edges

– coincidence
  • points, edges

Generation of an assembly model



Representation of Parts model

- Individual part
- Geometric modeling
- Attributes

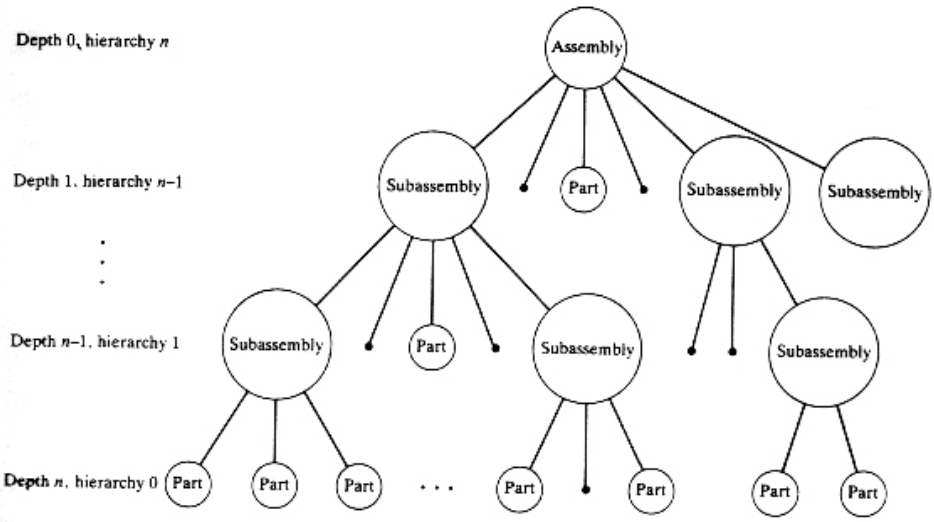
Simplified Representation

Large assembly models can serious stain the processor capabilities of the hardware system being used.

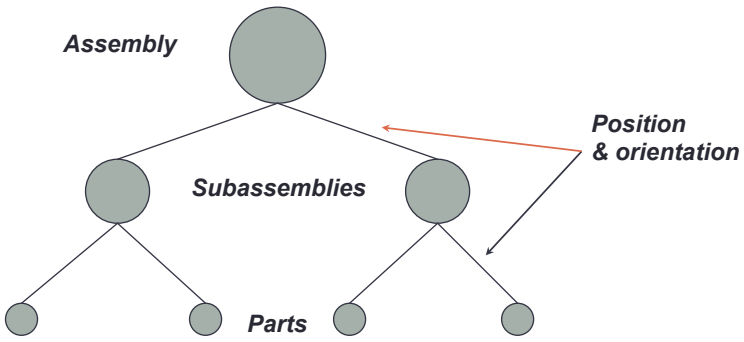
In these cases, many assembly modelers provide means for simplifying the assembly.

The use of instancing helps reduce complexity.

Assembly tree



Another tree

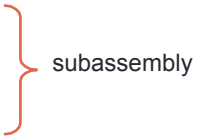


Assembly Hierarchy

- The assembly hierarchy defines relationships of parts to each other
- There can be multiple levels of sub-assemblies

Assembly Hierarchy Example

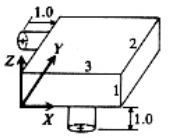
- Pump Assembly
  - Link assembly
    - offset link
    - offset link
  - Piping assembly
    - well casing
    - well seal
    - …
  - Handle assembly
  - ...



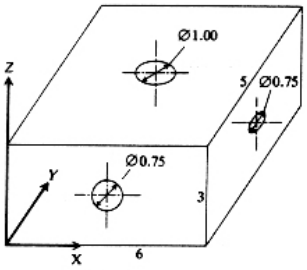
Positioning Parts in an Assembly

- Parts can be positioned by translating and rotating them into the right locations
- This requires careful measurement of relative locations, knowledge of coordinate systems, and entry of numerical values
- If position or dimensions of one part change, this has to be redone

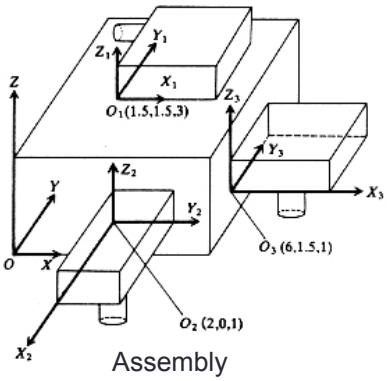
Assembly via GCS



Part B



Part A



Assembly

Assembly Constraints

- Constraints can be used to create permanent relationships between parts
- THEY use the same commands as 2D constraints
- Typical constraints:
  - two faces meet
  - axes coincident
  - two faces parallel at fixed distance
  - etc.

Mating condition

- Part coord. MCS (modeling coord)
- Base part: Datum
- Global CS
- Local CS
- Explicit position and direction vs. Mating conditions
- 4 x 4 homogeneous transformation matrix

Mating condition . . .

- Exploded view
- Instead of given transformation matrices
- User friendly: transformation matrices
- By changing conditions the assembly can be changed
- Assemblability
- Mating feature
- Types: against, fits, contact, coplanar
- fits: center lines are concentric

Mating condition . . .

- Mating condition = mating type + two faces
- Normal vector + one point on the face
- against: two normal vectors are in against directions
- fits: between two cylinders: center lines are concentric
- Against and fits allows rotation and translation between parts

Calculating positions from mating conditions

- Matrix calculation to find the position and orientation of each part
- Solve the system of equations
- Under constrained, Over constrained
- Over constrained: More eqs than number of free variables, eliminate redundant eq. or least means square.

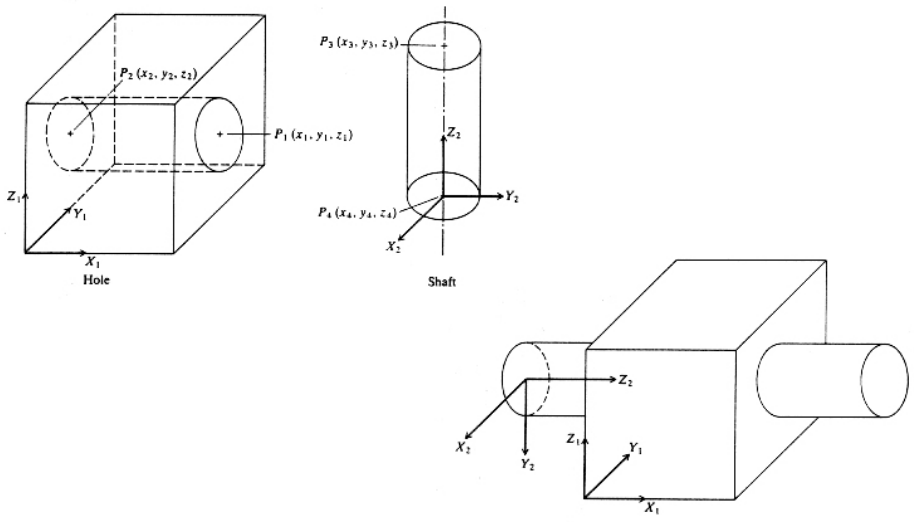
Positions from mating conditions. . .

- 3 mating conditions: against, fits, coplanar
- against : 2 normal vectors and 2 points
- 4 equations
  - n1 = - n2
  - n1 . (P1 – P2) = 0 : n directional component= 0

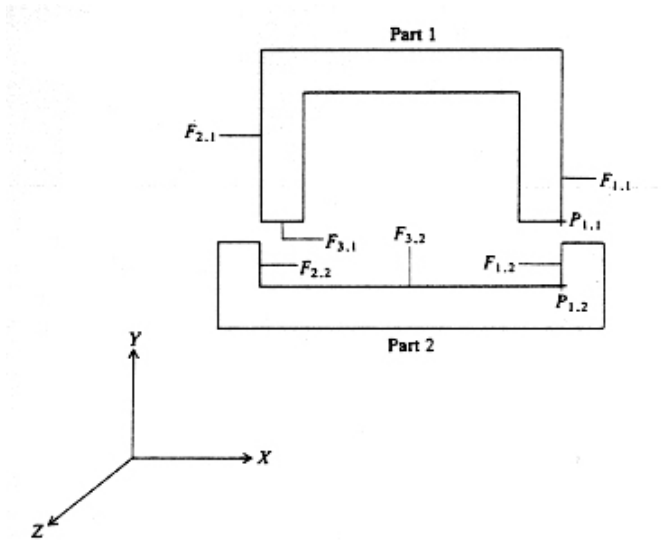
Against and fits

- Against and fits do not allow relative motion between parts: contact and tight fits
- Contact condition requires two points correspond: Rotation is allowed around the contact point
- First index = face number, 2nd index = part number: between two parts against conditions among 3 faces, contact condition between Ps

Fits condition



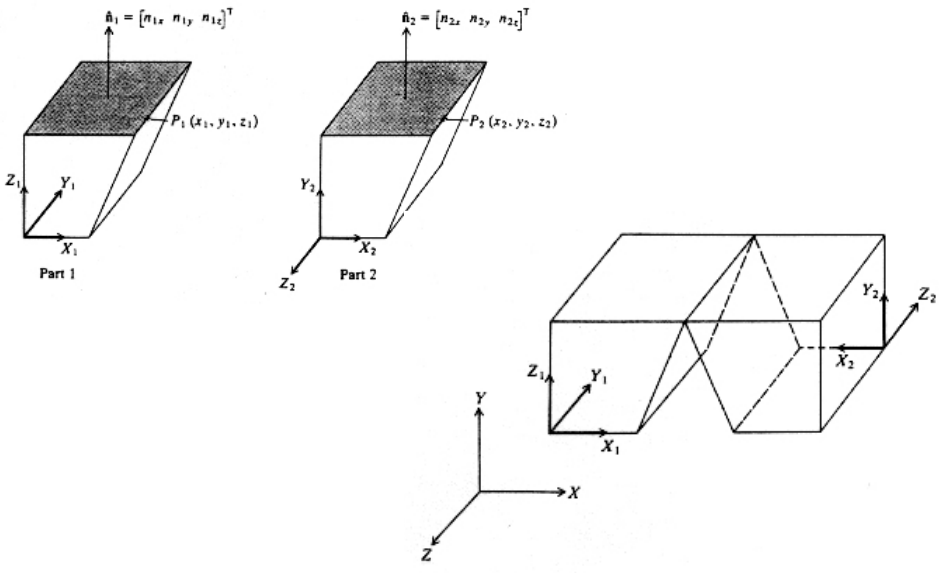
Contact condition



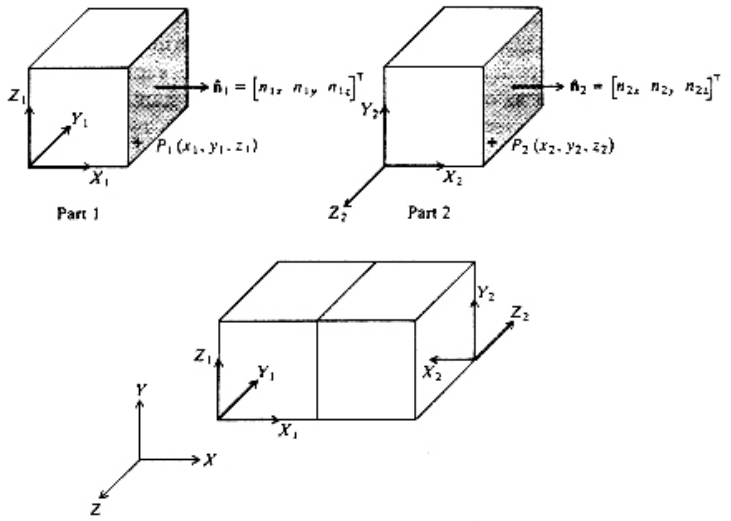
Interference fit

- Fits is clearance fit,
- tight fits is interference fit
- Coplanar: two normal vectors are parallel
- ‘Coplanar’ complements ‘against’

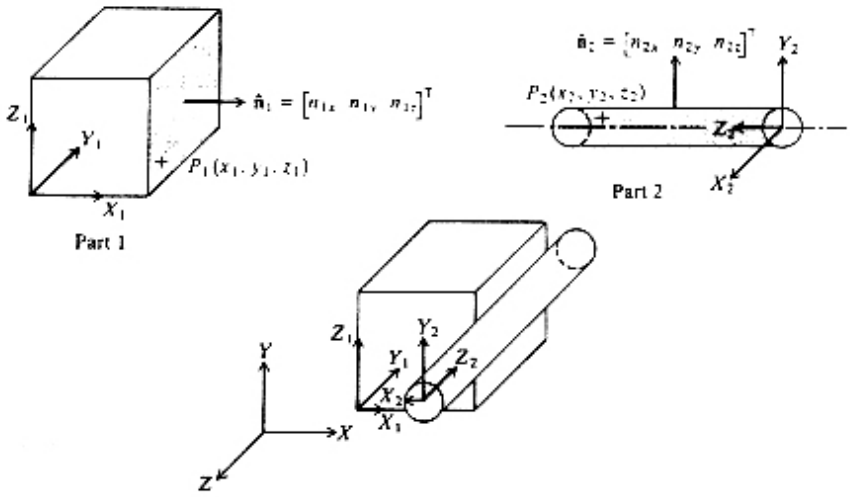
Coplanar condition



Against condition between two planar faces

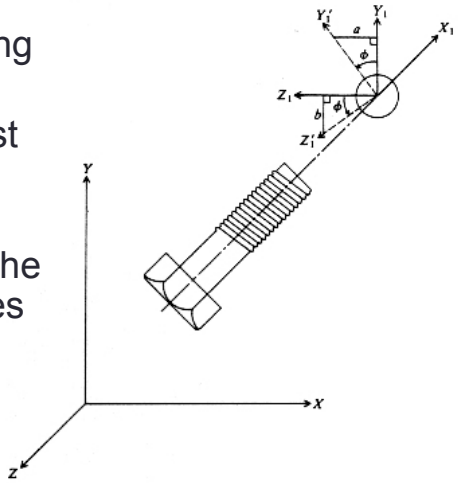


Against condition between a planar face and a cylindrical face



Fits and coplanar

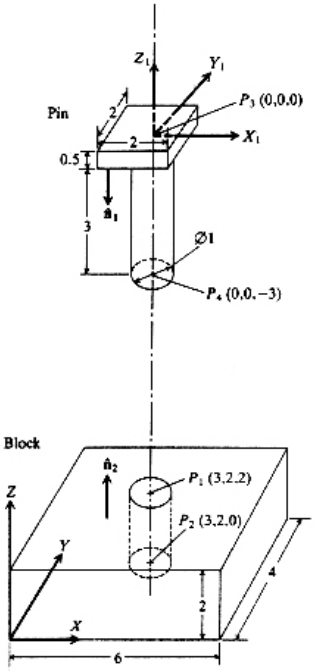
- Fits: Centerlines are on the same line
- 2 eqs are redundant among 6 eqs
- Coplanar: same as against except that normals are opposite direction
- If free rotation is allowed the matrix calculation becomes diverge:
- f = 0 : Constrain the free rotation



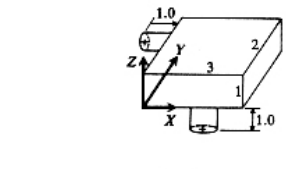
Solving the equation

- System of nonlinear eqs
- Least square : remove redundancy
- Newton-Raphson
- Jacobian matrix(∂Fi/∂xk, 1st partial derivative, approximating linear mapping)
- Linearization

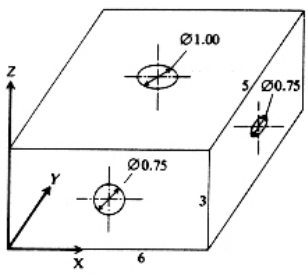
Pin and block Assembly

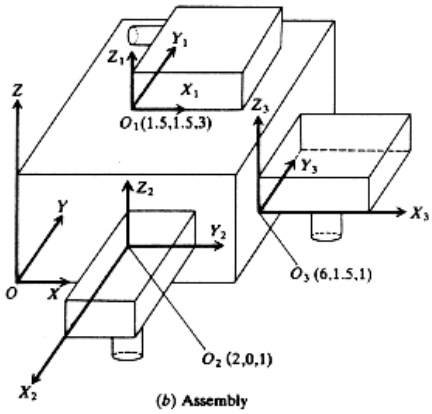


Assembly from instances



Part *B*





(*b*) Assembly

Representation Scheme

- An assembly database stores
  - the geometric models of individual parts
  - spatial positions and orientations of the parts in assembly
  - the assembly or attachment relationships between parts
- The inherent problem;
- how to assign assembly data interactively to build or develop the assembly

Representation Scheme

- Graph Structure
- Location Graph
- Virtual Link

Representation Scheme . . .

- The way the user provides the assembly data
  - the locations and orientations of the various parts.
  - their hierarchical relationship.
- 2 type schemes ;
  - utilize the WCS method : Graph structure
  - Location graph
  - utilize the mating conditions method -> Virtual link

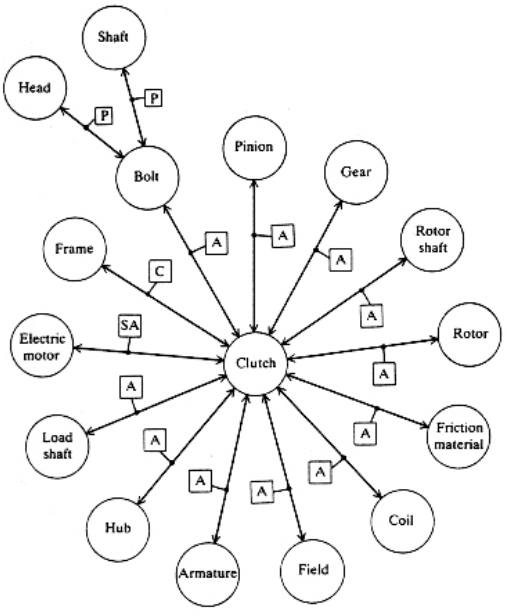
Graph Structure

- Use WCS (working coord. sys.) method.
- Each node represents an individual parts / subassembly.
- Arc (branch) represents relationship among parts
- P : 'part-of' relationship
  - Logical containment of one object in another.
- A : 'attachment' relationship
- C : 'constraint' relationship
  - physical constraints
- SA : 'sub-assembly' relationship

A : 'attachment' relationship

- rigid attachment: no relative motion
- non-rigid attachment
  - relative motion
  - parts cannot be separated by an arbitrarily large distance
  - useful in mechanisms analysis (joints)
- conditional attachment
  - related to parts supported by gravity, but not strictly attached

Graph structure of electric clutch



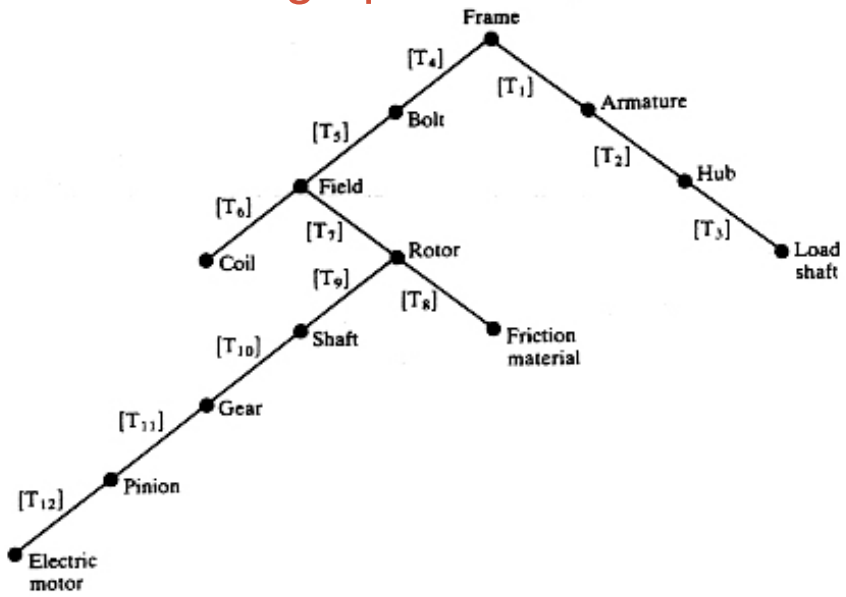
Location Graph

- WCS method
- Location of a part is a relative property.
- A chain of locations can be defined such that each location is defined in terms of [T] another part's coordinate system: Relative position.
- A set of these chains results in a location graph.

Location Graph . . .

- The 'frame' CS (coord. sys) is ;
  - the root of the location graph
  - taken as the global CS of the assembly .
- Related by the transformation matrix [T]

Location graph of electric clutch



Virtual Link

- Mating conditions.
- Graph structure + location graph requires the user to input transformation matrix.
- Virtual link requires more basic information (mating conditions)

  used to calculate the transformation matrices
- Based on the concept of virtual link.

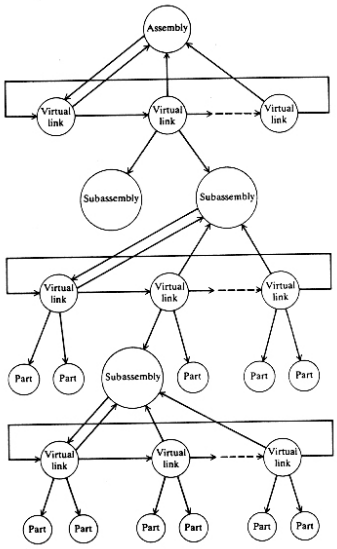
Virtual Link . . .

- Virtual link: defined as the complete set of information required to describe the type of attachment and the mating conditions between the mating pair.
- In the assembly data structure;
  - any mating pair is connected by a virtual link
  - if more than 2 parts are mutually related
  - several virtual links

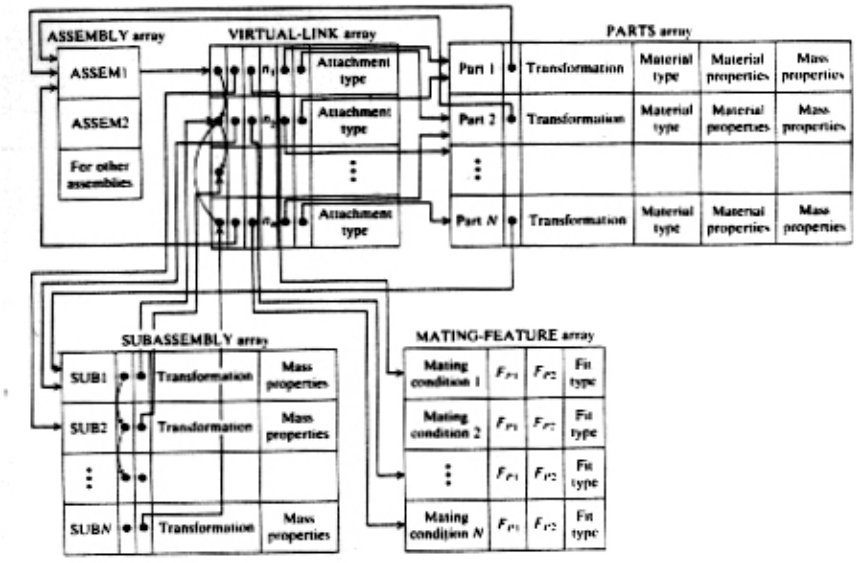
Virtual Link . . .

- Top node
- The leaf nodes are the parts of the assembly
- The geometric data for each one of these parts (leaf) are connected to these leaf nodes via pointers in the data structures

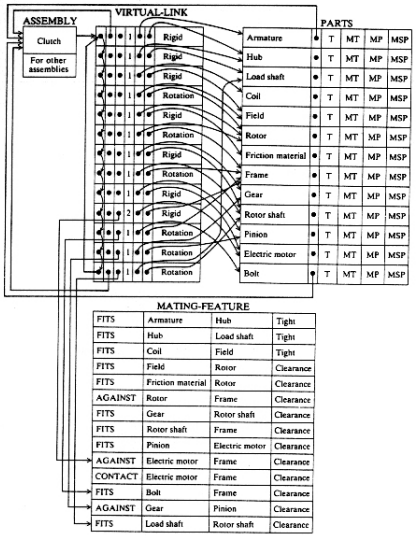
Assembly graph structure based on virtual link



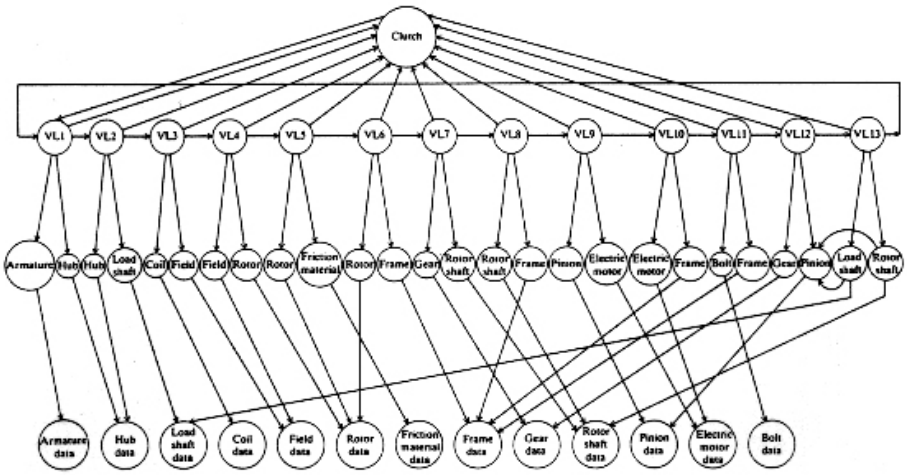
Data structure based on virtual link



Data structure of clutch based on virtual link



Graph structure of electric clutch based on virtual link



Generation of Assembling Sequences

- Precedence Diagram
- Liaison - Sequence Analysis
- Precedence Graph

Generation of Assembling Sequences

- In most assembly, there are multiple assembly sequence: Not unique
  => must decide on the most optimum sequence
- Assembling sequences of cars, ships, missile, Lego blocks

Assembly sequence affects

- difficulty of assembly steps
- need for fixture
- potential for parts damage during assembly and part mating
- ability to do in-process testing
- occurrence of the need for reworking
- time of assembly
- assembly skill level
- unit cost of assembly

Assembling Sequences . . .

- Trial-and error is practically impossible because;
  - large # of valid sequences
  - minor design changes can drastically modify the available choice of assembly sequences: design for production
- Few techniques: manual vs. algorithmic

Precedence Diagram

- Designed to show all the possible assembly sequences of a product.
- Each individual assembly operation is assigned a number.
- Diagram is usually organized into columns
  - in column I : the base part

Precedence Diagram . . .

Many operations

⇒ impractical on a single assembling machine

=> treat as sub-assemblies

⇒ easily studied & evaluated

multiple assembling machines can be used.

Precedence diagram

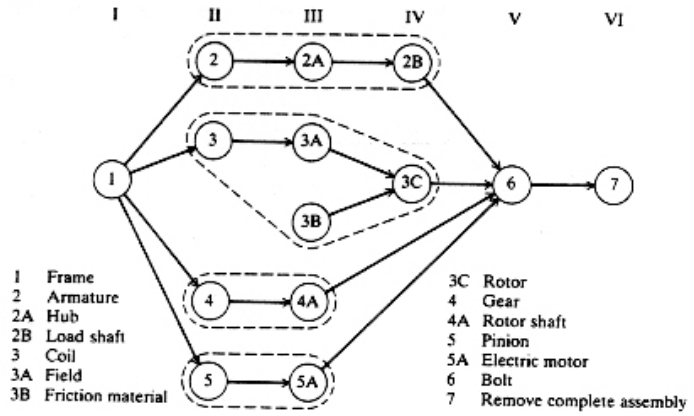


Diagram based on clutch individual parts

Precedence diagram

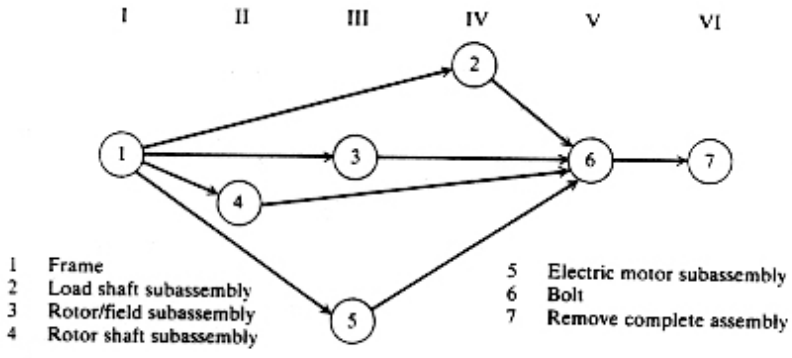


Diagram based on clutch subassemblies

Liaison - Sequence Analysis

Use precedence relations;

in precedence diagram: engineer generates possible sequence directly

in liaison method: asks a series of questions to engineers about mating condition and precedence.

=> generate sequences: manually or algorithmically

Liaison - Sequence Analysis . . .

Develops all possible assembly sequences in 2 steps ;

1) characterize the assembly by a network (liaison diagram)

2) generate all possible assembly sequences and represent them in liaison-sequence graph.

Liaison - Sequence Analysis . . .

Node: represent parts

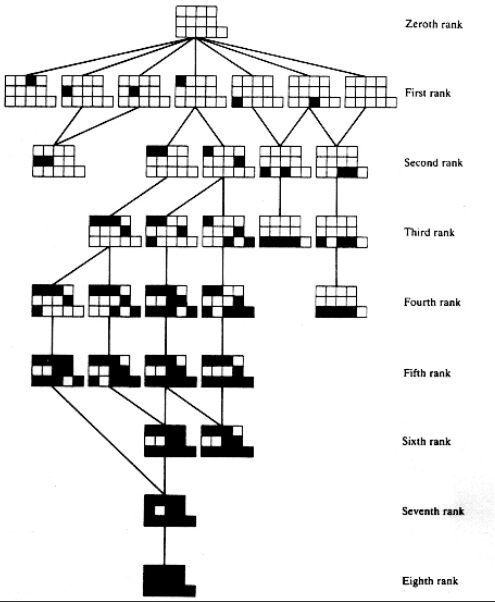
Line: represent any mating conditions between parts.

The liaison sequence diagram and list are not unique.

Determine the precedence relationships between liaisons by asking and answering.

Generate assembly sequences.

Liaison-sequence graph



Precedence Graph

Fully automatic

Based on virtual-link data structure

Requires the mating conditions as input.

Assembly sequence is generated with the aid of interference checking.

1) input mating conditions.

2) generate mating graph

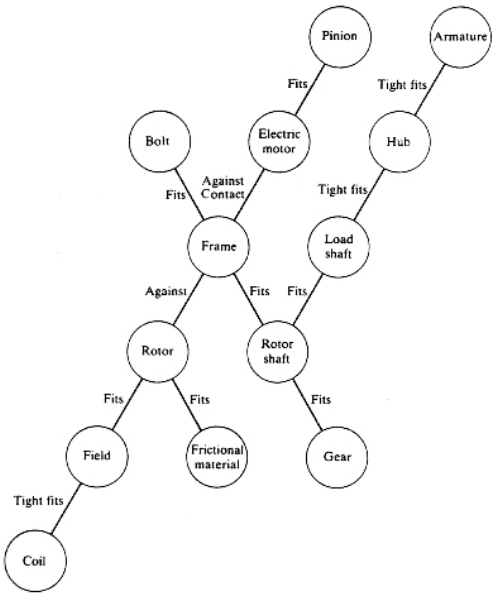
Precedence Graph . . .

identify the part that is connected to the largest # of parts by virtual links => the base part.

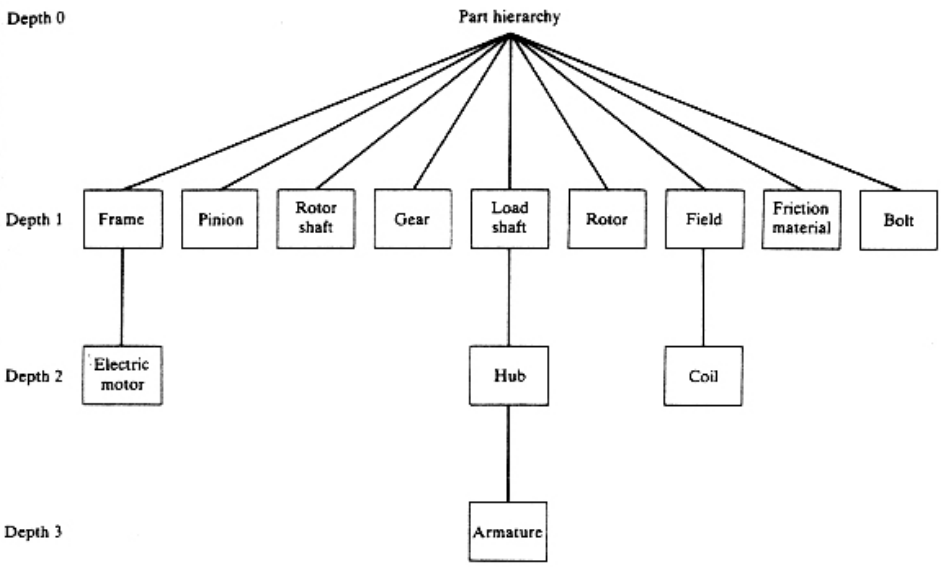
gather all the parts directly connected to the base part by virtual link.

- 1st step: a part hierarchy is developed based on the type of virtual link.
- 2nd step: develop the precedence graph with interference checking

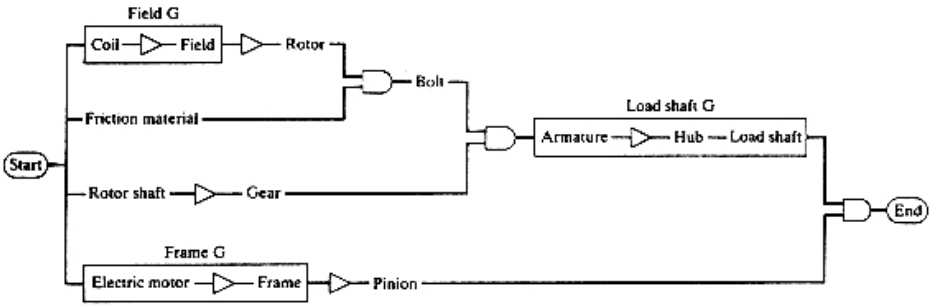
Mating graph for clutch



Part hierarchy of clutch assembly



Precedence graph of clutch assembly

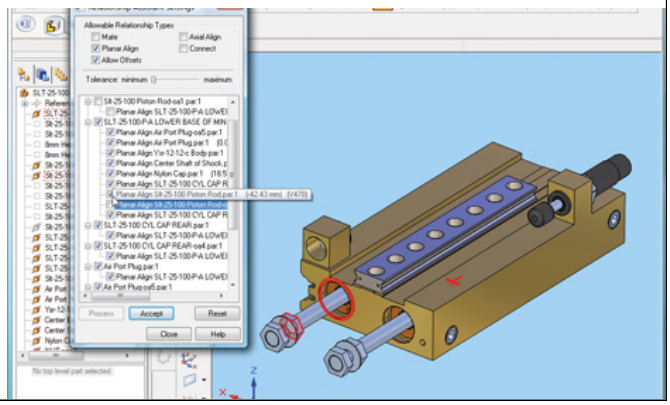


Assembly Analysis

Analysis tools ;
- generation of assembly drawings
- exploded views of assemblies
- shaded images of assembly
- cross sectional views
- mass property calculations
- interference checking
- kinematics & dynamic analysis
- finite element analysis
- animation & simulation

Intelligent Assembly Modeling & Simulation

The goal of IAMS is to avoid this expensive and time-consuming process by facilitating assemblability checking in a virtual, simulated environment.



Intelligent Assembly Modeling & Simulation