Chapter 2
Data Representation of Machine Models

2.1 Introduction

As we seek to understand the role and the capability of computers in design and analysis of machine tools, two overriding trends may be perceived: the computer graphics which is used for describing machine tool models being designed and analysed, and the coming integration of the information in the databases needed to automate design and analysis activities.

Works on computer graphics initiated almost 50 years ago. Serious efforts have been devoted since then to integrate the graphical interaction with the programs for designing physical systems, such as aircrafts, automobiles, buildings, ships as well as machine tools. It has been recognised that the computer is one kind of representational medium in the same way as drawings. Instead of representing the drawings and incorporating their inherent limitations, the computer can directly represent the target of the representation—the 3D physical system being designed and analysed. In such a computer representation, a single model can ideally depict the information that normally requires many drawings, as well as pages of specifications and engineering data.

Although the engineering drawings of machine tools have served us well in the past, current efforts towards the integration between computer-aided design (CAD) and computer-aided engineering (CAE) have shown that they have serious limitations as the means of geometric object description or definition. The problem is essentially that the drawings are understandable by humans but not by computers. Therefore, it is necessary to establish a suitable data representation and a common database of machine tools for both design and analysis.

To integrate CAD and CAE for machine tools, it is evident that industries must replace the traditional carrier of geometrical information—the drawing—with a computer-based information-carrying form (data representation) that is capable of supporting both the design and the analysis. To automate operations in these areas, this form of data representation must contain information that, being complete, consistent and accurate, is capable of supporting the application programs automatically.

L. Wang, *Dynamic Thermal Analysis of Machines in Running State*,
DOI: 10.1007/978-1-4471-5273-6_2, © Springer-Verlag London 2014

To achieve the integration and the automation mentioned above, the solid modelling with suitable data representation seems to be key.

In this chapter, solid modelling methods for three-dimensional objects are first classified, and then the data representations for machine tool models are introduced. Finally, a database of primitives is established based on an established data structure.

2.2 Classification of Solid Modelling Methods

Contemporary geometric modelling systems are concerned primarily with the geometry. They provide means for defining the shapes of components and sometimes allowable shape variations (tolerances), for positioning component representations to define assemblies, for calculating properties (appearance, mass, etc.) and for generating manufacturing-process data such as NC (Numerical Control) programs.

Essentially, the integration between CAD and CAE necessitates the use of some means of object definition other than the drawing. One such means relies on the construction of a common database using known mathematical terms in the representation of the object. Recent CAD systems possess the capability to define objects in three dimensions. This allows a designer to develop a full three-dimensional model of an object in the computer rather than a two-dimensional illustration. The computer can then generate orthogonal views, perspective drawings and close-up details of the object. Developments in this area include three-dimensional wire-frame, surface and solid modelling techniques [1]. Only the solid modelling technique is classified here due to its popularity and wide adoption in today's CAD/CAE systems.

Solid modelling is distinguished by the use of valid and unambiguous representations of solids. A solid model will involve both surface and edge definition of an object and will furthermore embody a recognition of volumetric details.

Two basic approaches that are used most frequently to solid modelling are constructive solid geometry (CSG) and boundary representation (B-reps) [2].

Constructive Solid Geometry This is a very direct and effective way to construct a solid object model. An object is modelled by a collection of primitives and a set of transformations and Boolean operations. The primitives, such as cuboid, cylinder, cone and sphere, etc., are parameterised. The transformations in this method include translation, rotation and scaling, and are used to define the positions, orientations and other features of the primitives. The Boolean operations, union, difference and intersection, are used to combine the primitives. In CSG, the final shape of a component is described and maintained internally by a tree structure of simpler shapes of primitives.

Boundary Representation This approach keeps a list of the faces, edges and vertices of a model together with the topological and adjacency relationships between them. In this case, the topology is used to determine the set of edges which constitute the boundary of a particular face or which meet at a specific vertex. In other words, a solid is represented by a finite number of bounded faces, each of which is

represented by a set of directed edges that bound it, and each edge is represented by two vertices. Each vertex is defined by a coordinate triple. Several kinds of regular surfaces can be used as the basic face elements for describing the solid object. These include planar surfaces (polygons) and analytical surfaces, such as cylindrical, conical and spherical surfaces.

Both methods (CSG and B-reps) have their relative advantages and disadvantages. In CSG systems, it is relatively easy to construct a topologically correct and precise solid model from the available library of primitives. It is compact in its storage requirements, but slow at producing images. On the other hand, a B-reps system gives designers more freedom in building complex models but the validity of the models could be destroyed in the process. It is also more expensive on memory. Table 2.1 gives the schematic illustrations of CSG and B-reps.

Four other unambiguous schemes for representing solids often in conjunction with CSG or B-reps schemes can be summarised as follows according to Bin [2], Meier [3] and Compton [4], which are also schematised in Table 2.1.

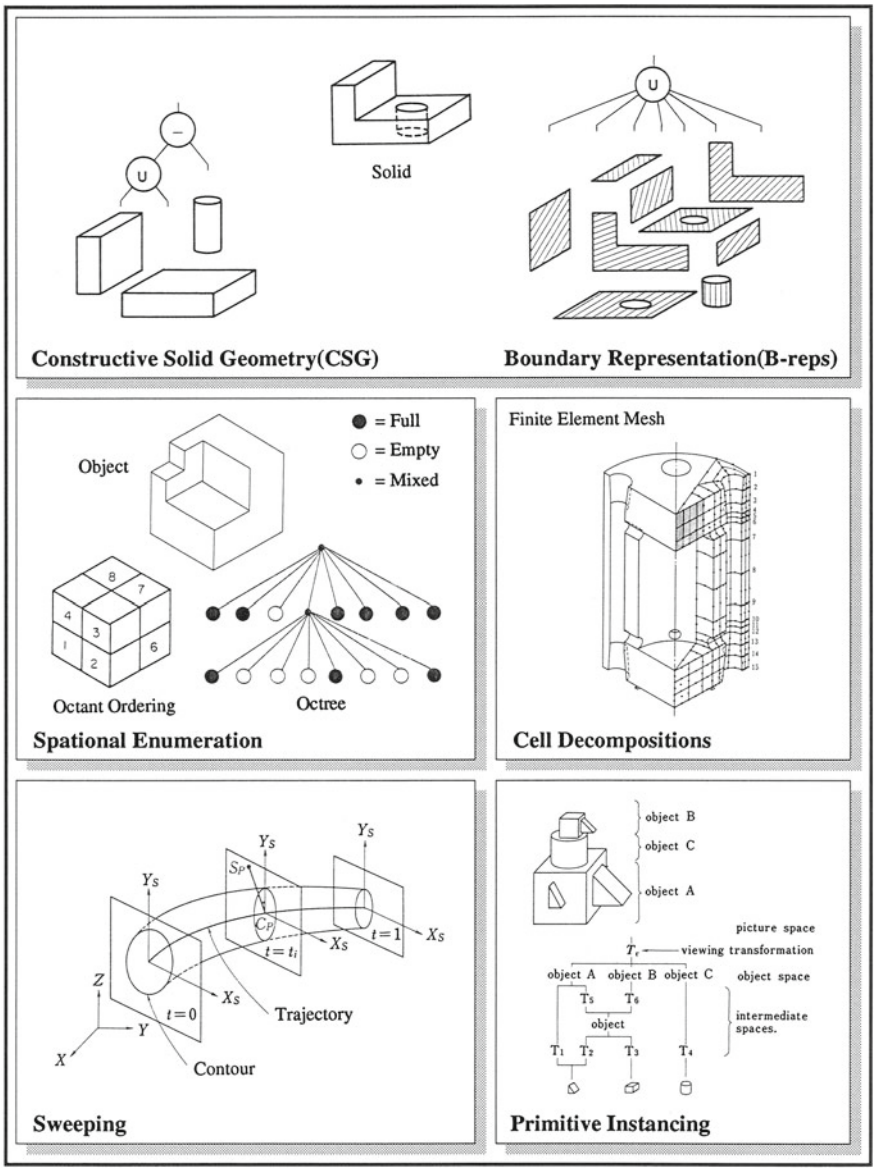
Spatial Enumeration A solid is represented (usually approximated) as a union of quasi-disjoint box-shaped cells 'filled with matter'. The cells may be of uniform size or of varying sizes if generated by recursive binary spatial subdivision. Enumerations of the latter type may be organised as logical trees, called quadtrees in two dimensions and octrees in three dimensions.

Cell Decompositions A solid is again represented as a union of quasi-disjoint cells, but now each cell may have a distinctive shape, provided that it is homeomorphic to a sphere. Triangulations are the simplest form of cell decomposition, and finite-element meshes are the most widely used engineering embodiments.

Sweeping A solid is represented as the spatial region traversed ('swept-out') by either an area or a solid moving on a spatial trajectory. There are two kinds of simple sweep representations: translational and rotational. The first is defined by a 2D contour and a straight trajectory along which it moves. The second is defined by a 2D generatrix and a rotational axis usually coplanar with the generatrix. Although sweeping is central to modelling motional processes such as machining and robotic assembly, there are many open mathematical and computational questions surrounding it.

Primitive Instancing This is a formalisation of the family-of-parts concept. A solid is represented as a particular member of a family by supplying appropriate numerical parameters to a family-specific collection of formulas for displaying members of the family, calculating their mass properties and so forth. Primitive instancing does not allow the combining of representations to create new or more complex objects. It is also difficult or even impossible to derive geometrical and topological properties directly from such schemes.

Table 2.1 Classification of well-known solid modelling methods



Most commercial solid modelling systems use a hybrid approach, converting the CSG working model into a B-reps definition, and store both models in the computer memory. The CSG might then be used for mass property calculations and the B-reps for downloading edge data for producing 2D drawings.

Table 2.2 Commonly used solid modelling systems

Modeller	Country	Yentor	Representation
CATIA	USA	IBM	B-reps
SHAPES	USA	Draper Lab.	CSG
PADL-1,2	USA	Rochester Univ.	CSG
TIPS	Japan	Hokkaido Univ.	CSG
GDP/GRIN	USA	IBM	CSG
SYNTHAVISION	USA	MAGI	CSG
GMSOLID	USA	General Motors	CSG,Sweep
U.M./BORKIN	USA	Univ. Michigan	CSG,Sweep
GLIDE	USA	Carnegie-Mellon Univ.	CSG,Sweep,EOP
EUCLID	France	Matia Data vision	B-reps
CADD	USA	MCAUTO	B-reps,Sweep
CATSOFT		CATRONIX	CSG
COMPAC	FRG	T.U./Berlin	CSG,Sweep
DESIGN	USA	MDSI	B-reps
BUILD-2	UK	Cambridge Univ.	B-reps
GEOMOD-II		SDRC	B-reps
MEDUSA	UK	CIS Ltd.	CSG,Sweep
UNIS-CAD	FRG	SPERRY UNIVAC	B-reps
GEOMAP	Japan	Univ. of Tokyo	CSG
UNISOLIDS	USA	MCAUTO	CSG
PROREN-2	FRG	Univ. RUHR	CSG,Sweep
SOLIDESIGN		Computer Vision	B-reps
ROMULUS	UK	ShapeData Ltd.	CSG,Sweep,EOP
DDM-SOLIDS		CALMA	B-reps
GEOMED	USA	Stanford Univ.	EOP
ICAD/CAE	Japan	Kobe Univ.	CSG,Sweep

EOP Euler Operations

Several well-known contemporary solid modelling systems are summarised in Table 2.2. *Build-2* and *Glide* are experimental prototypes, and *PADL-1* was designed mainly to demonstrate new algorithms and to serve as an educational and research tool. Most of the other systems in the table are either ready or thought to be nearly ready for industrial use [1]. *Build*, *Design* and *Romulus* rely internally on boundary representations. They are either created by direct manipulation of such low-level entities as faces and edges, or by means of Boolean and/or sweeping operations. *PADL* systems and *GMSolid* rely primarily on CSG both as an input and as an internal representational medium, but also maintain a boundary representation derived from CSG and therefore guaranteed to be consistent with CSG.

The most important characteristic of solid modelling systems is their ability to support (in principle) any geometric application because they are based on unambiguous (i.e. formally complete) representations.

Finally, the merits and limitations of solid modelling are clarified as follows:

Merits: Solid modelling provides the benefits of designing in 3D, a far more natural mode of expression. It can be used at many different stages in the design and analysis of a machine tool. At the conceptual design stage, it can provide a visual aid, possibly replacing the prototype.

Three-dimensional solid modelling allows sectional views at any position or angle, and exploded views are also possible. The 3D geometry can be transferred for structural stress analysis, dynamic response studies and for the generation of numerically controlled machining data.

Solid modelling also allows for interference checking; design errors which could be catastrophic at the machining or assembly stage of manufacturing can be pre-empted by simulation of the dynamic motions or assembly.

Limitations: In view of the complexities of the geometries generated using the three-dimensional modelling, it usually takes longer to construct a machine tool model in three dimensions than to produce the equivalent 2D views necessary for conventional manufacturing drawings. Furthermore, the computational demands of solid modelling can significantly reduce the performance of the system.

The use of a colour-shaded image is an important facet of the three-dimensional modelling. However, if one requested view does not quite show the details required, then more seconds of computer processing time may be needed before seeking the next view.

The limitations mentioned above can be overcome by a suitable data structure of machine tool models and a primitive library with characteristics of machine tool structures.

2.3 Representation of Machine Tool Models

Machine tool models are represented as combinations of primitives by adopting the CSG modelling technique in this book. Since most parts of machine tools are composed of planar polyhedra and bodies with simple curved surfaces, such as cylinder, cone and sphere, a machine tool-oriented primitive library should be established. Before this work is done, both geometrical and topological representations of geometric components, such as vertex, edge and face, etc., should be discussed first, since individual primitive is defined based on these representations.

Three kinds of geometric components being used here are defined as follows to clarify their constructive roles:

- A *face* is a set of points contiguous in two (not necessarily Euclidean) dimensions.
- An *edge* is a set of points contiguous in one (not necessarily Euclidean) dimension.
- A *vertex* is a bounding point of one or more edges.

Note that a *point* is different from a *vertex*; it is location in space defined by a triple $[X, Y, Z]$. The *space* is considered in its usual Euclidean form as an infinite point set, contiguous in the three dimensions of X, Y and Z.

2.3.1 Geometric Representation

Planar Polyhedra

Classical Cartesian coordinate geometry provides a number of coding schemes for representing the purely geometric aspects of planar polyhedra. Most commonly, a vertex can be represented by a point in coordinate space, a face by a plane equation and an edge by a pair of equations which specify a line, as in Fig. 2.1.

For complex manipulations and transformations in three dimensions, it is common to use a mapping of the objects into the homogeneous representation [5] in four dimensions. The homogeneous representation used in this book provides a more uniform formulation, and allows some extra degrees of freedom that are extremely convenient in numerical computation for dealing with the potential overflow, underflow and the truncation problems that can arise from the degenerate cases. Another advantage of homogeneous coordinates is that the extra degree of freedom in the plane representation allows one to distinguish the inside and outside of the surface. For example, if p is a 4-element column vector representing a plane, then the point represented by the vector V will be on the inside or the outside according to whether $V.p < 0$ or $V.p > 0$.

Topological element	Geometric type	Minimal representation	Number of values	Homogeneous representation	Number of values
vertex	point	(x, y, z)	3	$V = [x\ y\ z\ w]$	4
face	plane	$ax + by + cz + 1 = 0$	3	$P = \begin{bmatrix} a \\ b \\ c \\ k \end{bmatrix}$ $V.p = 0$	4
edge	line	$x = \frac{y - y_0}{a} = \frac{z - z_0}{b}$	4	$L = \begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ l_{21} & l_{22} & l_{23} & l_{24} \end{bmatrix}$ $V = [t\ l]\ .L$	8

Fig. 2.1 Mathematical representation of geometric components

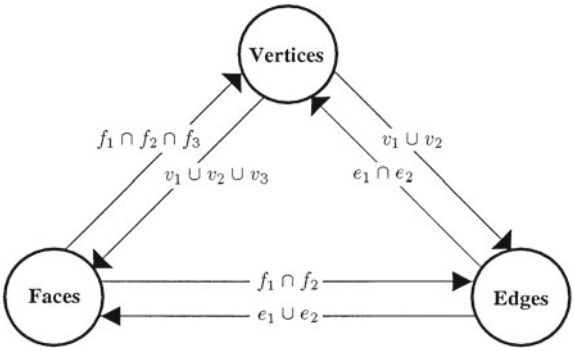


Fig. 2.2 Mapping among geometric representations

Curved Surfaces

It is always possible to approximate a curved surface using a number of planar faces to an arbitrary precision. For dynamic thermal analysis, an approximation is quite sufficient and an exact surface description is not needed. This approximation can be camouflaged in renderings and displays using various shading algorithms that give the graphical illusion of curvature without an explicit representation of the exact surface.

The approximation for curved surfaces is limited to certain classes of simple surfaces, such as cylinders, cones and spheres, which are considered as necessary for solid modelling of machine tools in this book.

Mappings Between Geometric Representations

Geometric definitions of the planar face, the edge and the vertex can be derived directly from each of the others [6]. A vertex is the intersection of two (coplanar) edges or three faces. An edge is the intersection of two faces or joins of two vertices (when planar), and a face can be computed to contain two (coplanar) edges or three vertices, as shown schematically in Fig. 2.2. Similarly, in the case of curved surfaces, edges and vertices can sometimes be computed from their intersections, with the restrictions mentioned above. In the planar case, each one of the classes of the geometric components (vertices, edges and faces) is by itself a complete geometric representation. The use of all three together is extremely redundant. However, the minimum size representation is not necessarily the most convenient one and some redundancy is often justified.

The question of which kind of information should be stored must depend on the intended use of the system. The solid modelling system presented in this book does not justify any face, edge and vertex information being stored, but only parameters from which they may be computed. This will be discussed later.

2.3.2 Topological Representation

Representing the geometric information is only part of the problems of modelling 3D shapes of machine tools. Only if a polyhedron is guaranteed to be convex, will its geometric representation of the faces, the edges and the vertices define a unique topology and thus an unambiguous shape. In the case of concave objects, there is some ambiguity even with oriented planes specifying the inside and the outside. Thus, the other part of the representation problems concerns the question of how the faces, the vertices and the edges are connected.

One approach being used in the solid modelling system is to avoid the issue by limiting the explicit representation to convex shapes. In such a system, the concave objects are represented as a combination of the convex objects, joined at 'virtual' faces. Within the topology of a polyhedron, there are nine classes of relationships among pairs of the three types of the geometric components (vertex, edge and face) [6]. They are shown, with a notation for characterising them, in Fig. 2.3. In notation $\{n_1 : n_2\}$, n_1 indicates a central geometric component, and n_2 a surrounding geometric component to connect with n_1. As with the geometric data, storing all of these relations is highly redundant. In fact, one type of relation is sufficient and all others can be derived from it. However, as some of the derivations are computationally expensive, it is often desirable to store more than one relationship. The question of which to store depends on the application. In the system used in this book, only

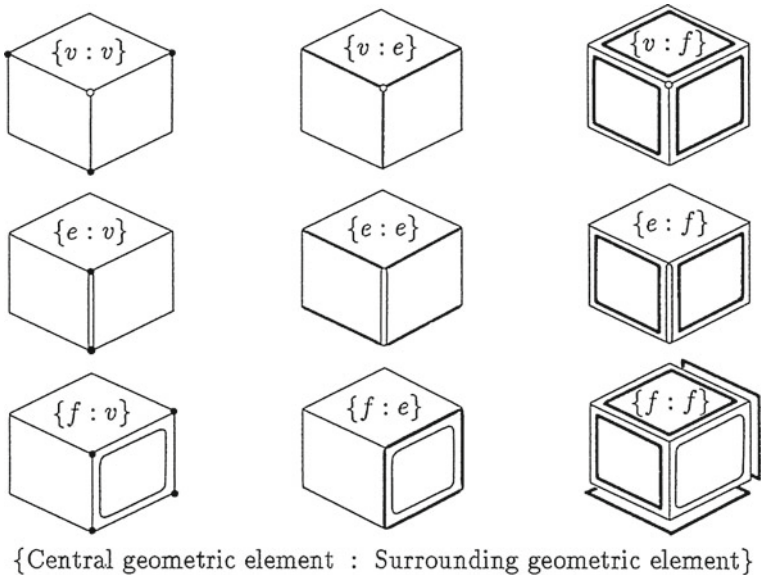


Fig. 2.3 The nine topological relations among geometric components

relations $\{v : v\}$, $\{e : v\}$ and $\{f : v\}$ are preserved so as to know how vertices are joined and how edges and faces are composed.

2.3.3 Construction of Shapes

Both the geometric and topological representations serve as internal representations of a solid modelling system. To specify shapes of a primitive to the system, only parameters of primitives (defined in detail later) are needed to be entered. These parameters constitute a data structure of the representations, in that the data represented by the parameters is expanded to incorporate the knowledge about the shapes of primitives. Often, there is little relationship between the parameters and the internal data representations since the internal data representations can be considered as a 'black box'. An example might be a cuboid parameterised by width, height and length from which the system will compute a large, sufficient set of shape information based on the internal data representations.

To define a new shape, the simplest and the most common way is as a linear transformation of an existing shape. The transformed shape is often known as an instance of the original, and may consist only of a translation and rotation, or may include scaling and shearing (change in size and proportion). Linear transformations affect the geometry of a primitive but not its topology.

The desire for generality, and for accessing parts of a shape description, has led the system designer to provide the means for constructing shapes from their elementary components: the faces, the edges and the vertices. The means for shape construction reflect the twin aspects of shape modelling—topology and geometry. Broadly, two basic approaches have been adopted, according to the sequence in which the topology and geometry is defined.

The solid modelling system starts with the definition of geometric components, either the vertex coordinates or the face equations, and subsequently define the topological connections between them. Parallel loft lines are interpolated between to define a face. The vertices are formed into rings clockwise to make faces and several faces are combined to form a body. The constructed bodies are checked to see that the faces match up and form a closed surface before a hidden-line removal or a shape operation is carried out.

The methods of shape definition described above may be viewed as hierarchically related, as shown in Fig. 2.4. The instances can be derived by transformation of an original primitive. Complex shapes, such as parts of machine tools, can be built by using CSG technique from a number of primitives and/or instances, while the primitive itself is constructed from the elementary geometric components connected by a topology. The definition of the topology in this system is more primitive than that of the geometry, for reason that it is much more common for many geometries to have a common topology than vice versa.

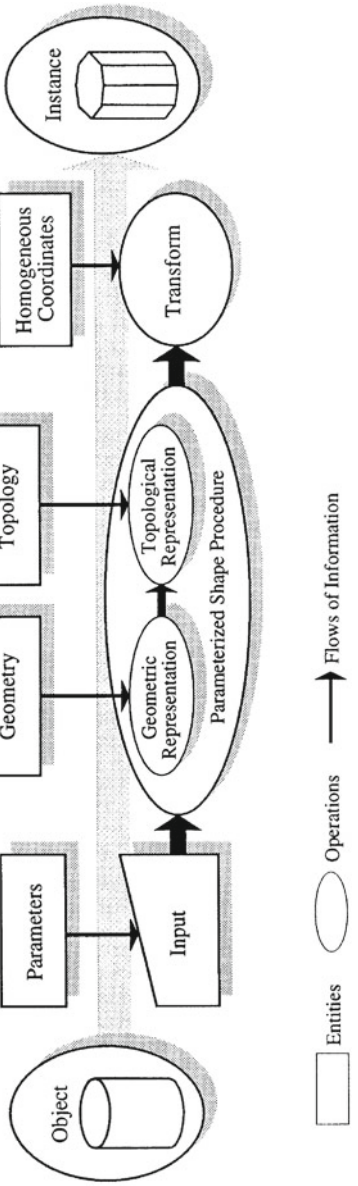


Fig. 2.4 Hierarchy of definition operations of primitives

2.4 Establishment of a Primitive Library

2.4.1 Data Structure

When machine tools and their parts are designed by engineers, the elementary design elements are not the geometric components (faces, edges and vertices), but the simple basic volumes called primitives in this book. The machine tools and their parts can be considered as being composed of a limited number of principal primitives. The set of the principal primitives is called a primitive library.

Since the CSG scheme is adopted to represent the machine tool models, it is necessary to establish a suitable primitive library in which each primitive can be chosen as a meaningful instance for constructing a machine tool model. The geometric and topological specifications for each individual primitive in the library are important in terms of solid modelling of the whole structure of the machine tool. Generally, these specifications for each individual primitive can be represented by an appropriate data structure which can best preserve all the necessary information.

The data structure employed could be considered as structural specifications of the internal data representations (geometric and topological representation) discussed in the previous section, as well as in the previous research [7]. Care has been taken to separate the topological structure (face/dege/vertex connectivity) from the geometric information (e.g. surface equations and vertex coordinates) and to keep the one consistent with the other in this data structure. All the relationships between them are linked by using pointers which make the information extraction convenient.

Figure 2.5 illustrates the overall configuration of the data structure adopted. There are seven entities being considered:

EDGE VERTEX LOOP FACE CURVED_F SPLINE_F CONTROL_P

where SPLINE_F and CONTROL_P preserve the attributes needed for B-spline approximations of the free-form surfaces.

The topology of the geometric components within a primitive is schematically illustrated by arrows. The geometries of a primitive are mainly stored in VERTEX, FACE and CONTROL_P. Equations of surfaces stored in FACE make the interference checks among primitives easier and faster. Another benefit of the equations is that approximations of curved surfaces take place at application time, not at model definition time.

Since the data structure proposed is only used for definitions of the primitive library, it is also called a low-level data structure in this book. The data structure for product modelling of machine tools, a high-level data structure, will be introduced in the next chapter.

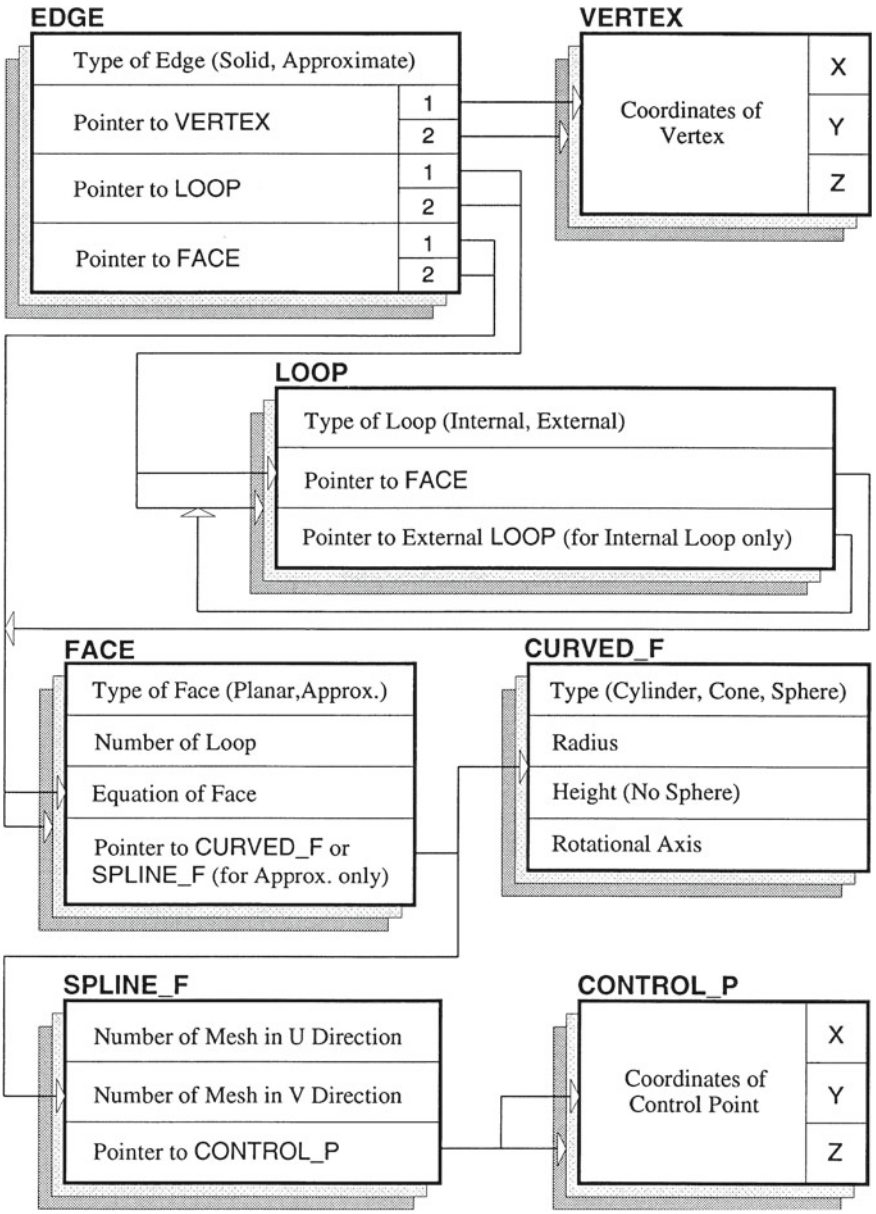


Fig. 2.5 A low-level data structure for primitive library

2.4.2 Definitions of Primitives

As mentioned above, the elementary design elements for solid modelling of machine tools and their parts are not the geometric components, but the primitives. The model of a machine tool is composed of a limited number of principal primitives in the primitive library by applying the CSG scheme. Therefore, definitions of the principal primitives are of vital importance for the solid modelling system. These definitions are carried out based on the data structure introduced previously to organise the topology and to store the geometries for each individual primitive. All the principal primitives in the library, considered in this book, are summarised as follows. Each of them, moreover, may include several variants.

1. Cuboid
2. Cylinder (*Circular Cylinder, Elliplical Cylinder, Prism*)
3. Cone (*Circular Cone, Elliplical Cone, Pyramid*)
4. Sphere (*Sphere, Spheroid, Ellipsoid*)
5. Swept Body (*Translational, Rotational*)
6. Box-type Primitive.

In order to create a certain primitive, one only has to enter a set of parameters which can best represent the primitive. Then, the computer will calculate the other information (geometric and topological) necessary for solid modelling of the primitive according to the low-level data structure. The question here is how to determine these parameters for each individual primitive, since the parameters not only should be able to identify each of the primitives completely but also should be minimum in quantity.

All of these parameters can mainly be divided into two types, in terms of their roles to play. They are:

- The parameters to represent the locations of primitives in Euclidean space, such as (O_x, O_y, O_z); and
- The parameters to represent the geometries of the primitives, such as width L_x, height L_y, length L_z and radius R, etc.

The definitions of the parameters for cuboid, cylinder, cone and sphere are given in Fig. 2.6 together with their schematic illustrations.

The parameters $O(O_x, O_y, O_z)$ in this figure indicate that an origin O of an object coordinates is located in the world coordinates by (O_x, O_y, O_z), representing the location of a primitive. However, only these four types of primitives are not sufficient for solid modelling of the machine tools and their parts. For those parts which have complex cross-sections, the sweep representation seems to be necessary. The sweep representations, both the translational and the rotational, are used in this book. To specify a swept body to the solid modelling system, a 2D cross-section (a contour for translational sweep or a generatrix for rotational sweep) should be defined first. Following the definition for the cross-section, a straight trajectory along which the cross-section moves and a value of sweeping distance are then required

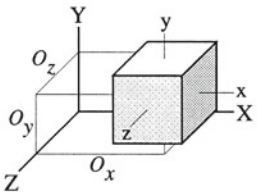
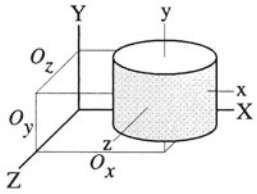
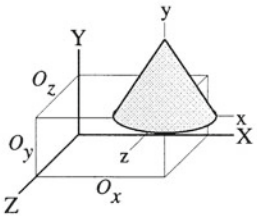
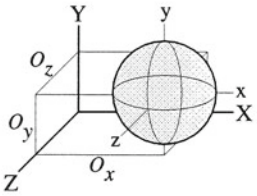
Type of Primitive		Definition of Parameters	Schematic Illustration
Cuboid		Origin: $O(O_x, O_y, O_z)$ Length: L_x, L_y, L_z	
Cylinder	Circular Cylinder ($R_x = R_z = R$)	Origin: $O(O_x, O_y, O_z)$ Radius: R_x, R_z Height: H Side Number: N (for Regular Prism only)	 In case of circular cylinder
	Elliptical Cylinder ($R_x \neq R_z$)		
	Regular Prism		
Cone	Circular Cone ($R_x = R_z = R$)	Origin: $O(O_x, O_y, O_z)$ Radius: R_x, R_z Height: H Side Number: N (for Regular Pyramid only)	 In case of circular cone
	Elliptical Cone ($R_x \neq R_z$)		
	Regular Pyramid		
Sphere	Sphere ($R_x = R_y = R_z = R$)	Origin: $O(O_x, O_y, O_z)$ Radius: R_x, R_y, R_z	 In case of sphere
	Spheroid ($R_x = R_y \neq R_z$)		
	Ellipsoid ($R_x \neq R_y \neq R_z$)		

Fig. 2.6 Definitions of parameters for primitives

for the translational sweep representation. For the rotational sweep, a rotational axis coplanar with the generatrix and an angle are required instead. The definitions for the sweep representations are given in Fig. 2.7. Note that the three constraints shown

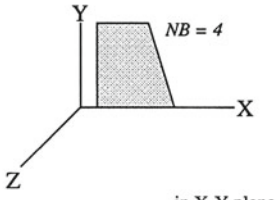
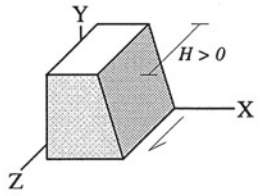
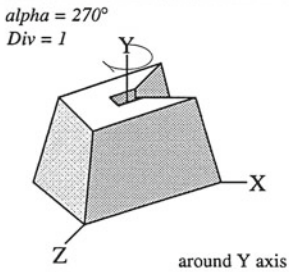
		Definition of Parameters	Schematic Illustration
	Contour or Generatrix	Number of Vertex: *NB* *(NB >= 3)*	
Sweep Representation	Translational Swept Body	Sweeping Length: *H* *(H > 0)*	
	Rotational Swept Body	Sweeping Angle: *alpha* *(0 < alpha <= 360°)* Dividing Number: *Div* *(Div >= 1, within 90°)*	
	Constraints	1. The vertices of the contour (or generatrix) must be given in clockwise-order. 2. The two consecutive sides must not be collinear. 3. The first two consecutive sides can not define a concave part of the contour (or the generatrix).	

Fig. 2.7 Definitions of parameters for sweep representations

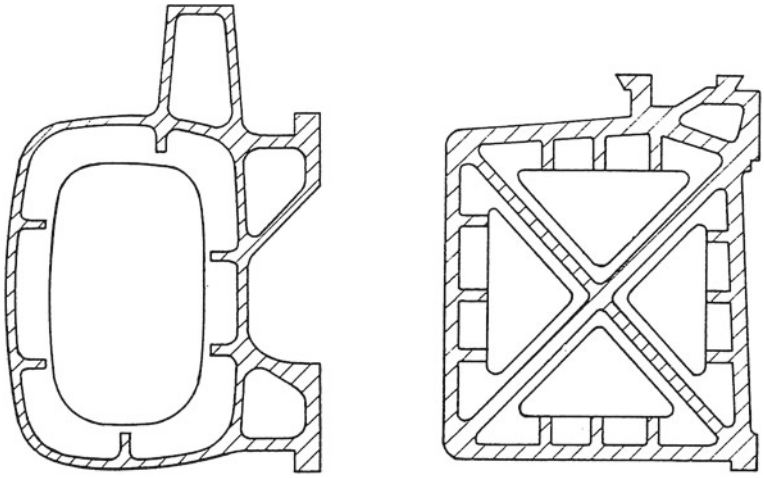


Fig. 2.8 Typical cross-sections of machine tool structures

in this figure should be observed firmly, due to the limitation of the chosen computer graphics.

The primitives defined above are solid, or 'filled with matter'. But most parts of machine tools are not always solid if looking into their internal structures. The layout of the internal structures, as well as the shapes and the sizes of the parts, must be so designed as to ensure the following: (1) the satisfactory conditions exist for the operation and maintenance of the machine tools, (2) the working stresses, deformations, deflections and displacements under working conditions remain within the specified limits and (3) the total weight of the structures and the weight distributions of the parts satisfy the technical and economic requirements. In consideration of these factors, the internal structures of the parts are usually designed in the way shown in Fig. 2.8 to satisfy the requirements of the stiffness, i.e. the resistance to deformation under load, and to minimise the total weight of the machines. However, it should be noticed that the stiffness-to-weight ratio and the actual total weight must be taken into consideration to keep the natural frequency of the machine tool outside the range of its spindle speed in order to prevent any chatter vibration.

Combining these defined primitives properly, the structures of machine tools and their parts can be modelled realistically. This is of importance to increase the accuracy of various analyses and simulations of the machine tool performance, and hence to obtain a much more reasonable structure of the machine tool.

Several samples of the principal primitives in the library are summarised in Fig. 2.9.

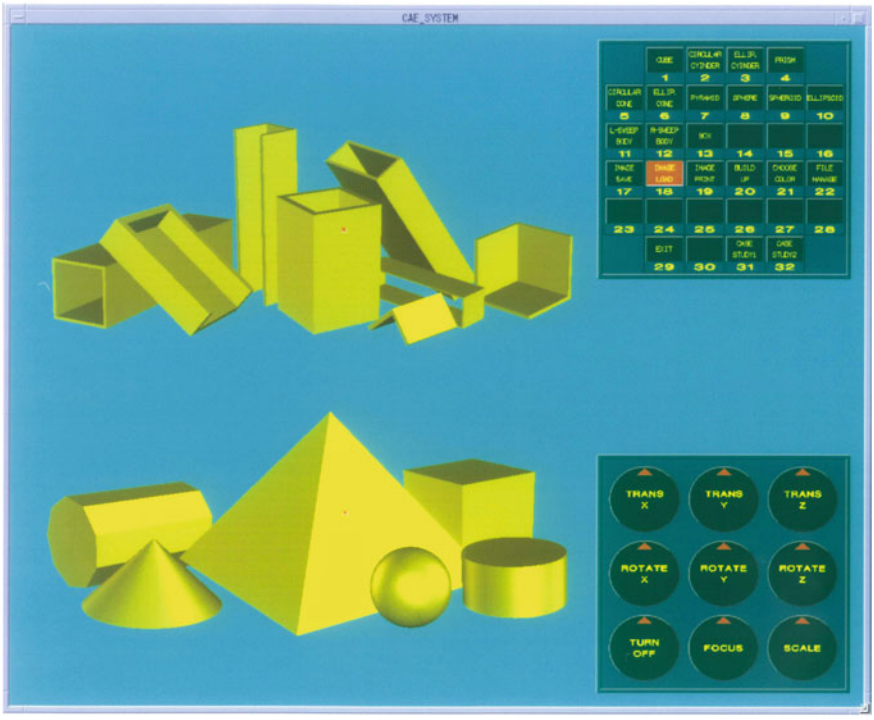


Fig. 2.9 Samples generated from the library

2.5 Concluding Remarks

In this chapter, both the solid modelling methods for three-dimensional objects and the data representing method for machine tool models were discussed. A new data structure was presented, and seven types of principal primitives were defined for the purpose of implementing a primitive library. The main topics discussed in this chapter can be summarised as follows:

1. Two basic approaches that are used most frequently to solid modelling are constructive solid geometry (CSG) and boundary representation (B-reps). The CSG scheme was adopted in this book.
2. Since machine tool models can be represented as the combinations of a small number of principal primitives by using CSG, a primitive library was established based on a specific data structure proposed in this book.
3. Most parts of machine tools are actually designed with thin-walled box sections. Therefore, box-type primitives can be added to the primitive library to provide a realistic way for solid modelling of machine tool structures.

References

1. A.A. Requicha, H.B. Voelcker, A historical summary and contemporary assessment. IEEE Comput. Graph. Appl. **2**, 9–24 (1982)
2. H. Bin, Inputting constructive Solid geometry representations directly from 2D orthographic engineering drawings. Comput. Aided Des. **18**(3), 147–155 (1986)
3. A. Meier, Applying relational database techniques to solid modelling. Comput. Aided Des. **18**(6), 319–326 (1986)
4. W.D. Compton, *Design and Analysis of Integrated Manufacturing Systems* (National Academy Press, Washington, DC, 1988), pp. 167–199
5. E.A. Maxwell, *General Homogeneous Coordinates in Space of Three Dimensions* (University of Cambridge Press, UK, 1951)
6. A. Baer, C. Eastman, M. Henrion, Geometric Modeling: A Survey. Comput. Aided Des. **11**(5), 253–272 (1979)
7. L. Wang, *A Study on Modeling System for Computer-Aided Engineering, Master Thesis, Graduate School of Engineering* (Kobe University, Japan, 1990). (in Japanese)

http://www.springer.com/978-1-4471-5272-9

Dynamic Thermal Analysis of Machines in Running State

Wang, L.

2014, XV, 128 p., Hardcover
ISBN: 978-1-4471-5272-9