Preface

The original intention of this edited book came from the discussion with Professor John Davies, the Chief Editor of the International Journal of Advanced Manufacturing Technology (IJAMT), on the proposition to publish a Special Issue on the applications of virtual and augmented reality technology in manufacturing in September 2002.

We were overjoyed that the call-for-paper of this Special Issue had attracted favourable responses from many top researchers in this field. Mr. Anthony Doyle, Commissioning Editor (Engineering) and Ms. Christiane S Notarmarco, Corporate Sales & Journals Development Manager of Springer-Verlag London Ltd., suggested that instead of having a Special Issue in IJAMT, the papers could be edited into a book that would fit well into the manufacturing engineering collection of Springer. We were quite excited about this invitation and immediately contacted the contributors. The response from all the contributors was very positive and the proposal for a book was submitted to Springer-Verlag for evaluation. The good news that the Springer Editorial Committee had approved the publishing of the book was conveyed to us in July 2003. As the original intention was to be a peer-reviewed Special Issue, and all the papers were either in the process of being reviewed or had already gone through the reviewing process, we informed the contributors that the quality of each paper, now each chapter, had followed the same standard of a rigorously peer-reviewed international journal.

We intend this book to be of interest to researchers, graduate students and practising engineers involved in the simulation, training, design and analysis of manufacturing problems using AR and VR techniques As VR and AR technologies are fast evolving, we certainly hope that this book can serve as a useful insight to the readers on the state-of-the-art applications and developments of such techniques at the time of compilation.

We are most grateful to the kind cooperation of all the renowned contributors who had promptly responded to all the questions and had painstakingly followed our requests for additional information. We would also like to thank Springer-Verlag for giving us this opportunity of publishing this book.

<div style="text-align: right">

S.K. Ong
A.Y.C. Nee

</div>

2 An Approach for Solid Modelling in a Virtual Reality Environment

Yongmin Zhong[1], Weiyin Ma[2]

[1]School of Computer Science, University of Windsor, Canada
[2]Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China

With today's Virtual Reality (VR) systems, it is difficult to directly and precisely create and modify objects in a VR environment. This chapter presents an approach for solid modelling in a VR environment. Solid modelling in the VR environment is performed precisely in an intuitive manner through constraint-based manipulations. A hierarchically structured and constraint-based data model is developed to support solid modelling in the VR environment. The data model integrates a high-level constraint- based model for precise object definition, a mid-level CSG/Brep (Constructive Solid Geometry/Boundary representation) hybrid solid model for hierarchical geometry abstractions and object creation, and a low-level polygon model for real-time visualization and interaction in the VR environment. Constraints are embedded in the solid model and are organized at different levels to reflect the modelling process from features to parts. Constraint-based manipulations are accompanied with automatic constraint recognition and precise constraint satisfaction to establish the hierarchically structured constraint-based data model and are realized by allowable motions for precise 3D interactions in the VR environment. The allowable motions are represented as a mathematical matrix for conveniently deriving allowable motions from constraints. A procedure-based degree-of-freedom combination approach for 3D constraint solving is presented for deriving the allowable motions. A rule-based constraint recognition engine is developed for both constraint-based manipulations and implicitly incorporating constraints into the VR environment. A prototype system has been implemented for precise solid modelling in an intuitive manner through constraint-based manipulations.

Keywords: *Virtual Reality; Solid Modelling; Constraint-based Manipulations; Constraint Solving; Constraint Recognition*

2.1 Introduction

The VR technology is regarded as a natural extension to 3D computer graphics with advanced input and output devices and it brings a completely new environment to the CAD (Computer-Aided Design) community. However, the integration of VR and CAD is not an easy task. At present, there are two methods for combining VR with CAD. In the first method, which is employed by most of the current VR systems (Sa and Zachmann 1999; Whyte et al. 2000), VR is only used as a toolkit for visualizing and analyzing CAD models. With this method, CAD models are first created by using CAD software, such as AutoCAD, UGII, ProE, etc., and the created CAD models are then imported into a VR environment for visualization and analysis. Some difficulties with this approach are as follows:

∉ The models are first created in the CAD systems by specifying the detailed dimensions while these dimensions are not precisely defined in the concept stage (Dani and Gadh 1997).

∉ Topological relationships and constraints between entities and parametric information are lost when transferring the models from the CAD systems to the VR systems (Berta 1999).

∉ To modify the models, one must return to the CAD systems to make the desired changes and re-import the revised models into the VR systems for verification (Gao et al. 2000).

The second method directly creates solid models in a VR environment by developing novel CAD systems (which are called VR-based CAD systems) (Gao et al. 2000; Stork and Maidhof 1997; Zhong et al. 1999). With this method, all the design activities are carried out in the VR environment. Users can intuitively create and modify 3D shapes through 3D direct manipulations, and visualize and analyze the design in the same system without any data transfer. The second method overcomes the major limitations of the first method. However, most of the existing VR systems only offer very limited tools for solid modelling, and lack sophisticated modelling and modification tools for creating complex solid models in a VR environment. Among others, the finite resolution of virtual objects without topological information is not suited to represent solid models for design purposes. The limited accuracy and reliability of 3D input and output devices also prevent users from precise design activities.

This chapter presents an approach for solid modelling in a VR environment. Solid modelling is performed precisely in an intuitive manner through constraint-based manipulations. A hierarchically structured and constraint-based data model is developed to support solid modelling in the VR environment. This data model integrates a high-level constraint-based model for precise object definition, a mid-level CSG/Brep (Constructive Solid Geometry/Boundary representation) hybrid solid model for hierarchical geometry abstractions and object creation, and a low-level polygon model for real-time visualization and interaction in the VR environment. Constraints are embedded in the solid model and organized at different levels to reflect the modelling process from features to parts. Constraint-

based manipulations are accompanied with automatic constraint recognition and precise constraint satisfaction to establish the hierarchically structured constraint-based data model, and are realized by allowable motions for precise 3D interactions in the VR environment. The allowable motions are represented as a mathematical matrix so that they can be conveniently derived from the constraints. A procedure-based degree-of-freedom (DOF) combination approach for 3D constraint solving is presented for deriving the allowable motions. A rule-based constraint recognition engine is developed for both constraint-based manipulations and implicitly incorporating constraints into the VR environment. A prototype system has been implemented for precise solid modelling in an intuitive manner through constraint-based manipulations.

2.2 Related Work

The brief survey first gives an overview of constraint solving, and then introduces the applications of direct manipulations in geometric modelling. Afterwards, the focus turns to the existing methods for solid modelling in a VR environment. Finally, current techniques for the integration of VR, constraint solving and direct manipulations in solid modelling are discussed.

2.2.1 Constraint Solving

Some of the major constraint solving approaches can be classified as follows:

∉ Numerical algebraic approach: This is one of the commonly used techniques for constraint solving (Sutherland 1963; Light and Gossard 1982). In this approach, all constraints are translated into algebraic equations and the instances of a geometric model are derived by solving these equations with numerical techniques, such as the Newton-Raphson iterative method and its refinement methods. The numerical algebraic approach is quite general and is capable of dealing with over constrained, consistent constraint problem but the convergence to a solution is not always guaranteed and the final solution depends on the choice of initial values.

∉ Geometric reasoning: Systems reported by Aldefeld (1988) and Ambler and Popplestone (1975) are based on a geometric reasoning approach. This approach employs artificial intelligence to perform the symbolic manipulations of constraints. It provides generic solutions. Nevertheless, it depends on the relativity of the parameters and is also computation-extensive. This approach is also referred to as the symbolic algebraic approach.

∉ Constraint propagation: In (Gossard et al. 1988; Khatib 1996), a procedural constraint propagation technique was adopted. The method allows a user to position new geometric elements relative to existing ones in terms of geometric constraints. The systems, however, require a user to construct geometric

elements in a very restricted manner and cannot handle under-constrained geometric elements.

∉ DOF analysis: Kramer (1991) proposed a technique called the DOF analysis that has significant computational advantages over the algebraic and geometric reasoning approaches. Using this technique, a sequence of operational transformation is automatically devised to satisfy each constraint incrementally.

There are also some other approaches for constraint solving, such as the graph-based representation and the rule-based method. In general, most constraint-solving systems deal with 2D constraints or kinematics problems mainly because of the complexity of constraint solving in 3D problems (Lhomme et al. 1998). Presently, there are especially few research work that focused on integrating 3D constraint solving with 3D direct manipulations.

2.2.2 Direct Manipulations

Direct manipulation has already been successfully applied in geometric modelling. For example, a user can control the geometry of an object by grasping and dragging operations using direct manipulation techniques and update the geometry continuously.

Bier presented a direct manipulation method named "snapping-dragging" for creating 3D objects using a 2D mouse (Bier 1986). This method combines three interactive techniques: grid-based interaction, alignment and interactive transformation. In this method, precise interactions are realized by snapping a 3D cursor and moving it to a set of points, lines, planes and spheres displayed on the screen. However, the desired 3D positions depend on a set of specified transformations. Furthermore, since menu interactions are often required, the interactions are very tedious and unintuitive. Emmerik also presented a method for direct manipulation of 3D objects using 2D input devices (Emmerik 1990). In this method, the direct manipulations of 3D objects are realized by manipulating the geometry trees of 3D objects. Gleicher developed the Brambler graphics toolkit that supports interactions using the differential manipulation technique (Gleicher 1993). The smooth manipulation of dragging an object is realized by the constraint that forces the objects to move towards the current position of the cursor.

In connection with direct manipulations and constraint solving, TWEAK is a constraint-based manipulator for editing 3D objects using 2D cursors (Hsu et al. 1997). It provides a toolkit for placing the vertices, planes and objects picked by a user. The manipulator is connected to a 3D geometric constraint solver, which ensures that the changes are consistent with the relationships between the geometric elements. Kwaiter et al. presented a geometric constraint system called LinkEdit that provides an interactive 2D tool to construct objects from rigid primitives and constrain them by several constraint types (Kwaiter et al. 1997). When an object is selected, an interactive constraint is added into the constraint graph and the constraint solver re-satisfies the constraint graph. When the object

is being moved, a series of local modifications are performed. When the object is released, the added constraint is deleted from the constraint graph.

2.2.3 Solid Modelling in a Virtual Reality Environment

The use of VR for CAD is not totally new. In the area of 3D modelling, one of the earliest systems was 3DM that allows users to interactively create simple geometric objects, such as cylinders and spheres, in the VR environment (Butterworth et al. 1992). 3DM includes several grid and snap functions. However, it lacks many other aids and constraints that are necessary to accomplish precise work. JDCAD also tackled many issues for interactive 3D objects modelling (Liang and Green 1994). Users could directly interact in a 3D space using a 6-DOF input device. However, only simple solids can be created in JDCAD. Dani and Gadh (Dani and Gadh 1997) presented a COVIRDS system for concept design in the VR environment. This system is based on design features and a geometric modelling kernel ACIS is used for their development. The precise interactions mainly rely on voice commands. Stork and Maidhof also reported some work on interactive and precise solid modelling using a 3D input device (Stork and Maidhof 1997). Precise modelling is realized using 3D grids, grid snapping and discretized dimensions. Constraint-based interactions are based on pre-defined rules for feature-based modifications. Although precise modelling can be ensured, the constraint-based interactions are too rigid for extensive use. Nishino et al. presented some results on gesture-based 3D shape creation (Nishino et al. 1999). A 3D modeller is developed to create complex shapes by combining the defined hand actions while precise interactions are not included. Gao et al. reported a method on constraint-based solid modelling in a semi-immersive VR environment (Gao et al. 2000). In this method, the manipulations to a primitive depend on some shape control points (SCP) on the primitive instead of the primitive itself, and a 3D mouse must be set to the SCP for manipulating the SCP. Furthermore, the SCP cannot sufficiently reflect the natural behaviours of the geometric elements of the primitive. Therefore, the interactions in the virtual environment are unintuitive and inconvenient.

There are also a few researches that focused on the integration of VR, constraint solving and direct manipulations. Fa et. al. reported some results on 3D object placement (Fa et al. 1993). Fernando et. al. further extended the results into a shared virtual environment (Fernando et al. 1995) and presented a software architecture of a constraint-based virtual environment (Fernando et al. 1999). The most important contribution of their method is the concept in constraining 3D direct manipulations through the allowable motions of an object being manipulated for precise locations and operations. Since the allowable motions are derived from some predefined rules that are related with the constraint types and geometric element types, they are difficult to be used extensively. Furthermore, only simple geometry and constraints are treated and simple solid models can be

created in the VR environment. Complex models are still created from CAD systems and then imported into the VR environment.

The authors also reported some preliminary results on precise solid modelling in a VR environment (Ma et al. 1998; Zhong et al. 1999). The article (Ma et al. 1998) reported some results on creating assemblies with embedded constraints between mating features through direct manipulations while (Zhong et al. 1999) reported some results on creating parts by features through direct manipulations. In general, the results reported in (Ma et al. 1998; Zhong et al. 1999) were at the initial stage and only the conceptual solid modelling framework in a VR environment was presented. There is still a lot of space for further development and improvement.

This chapter is based on the authors' previous work and presents the details on solid modelling in a VR environment. The goal of this research is to develop an intuitive 3D environment for solid modelling. A hierarchically structured and constraint-based data model is presented to support solid modelling in a VR environment. Constraint-based manipulations are elaborated for precise solid modelling. A procedure-based DOF combination approach for 3D constraint solving is presented to derive the allowable motions. Furthermore, a prototype system for solid modelling in a VR environment has been implemented to demonstrate the research work.

2.3 Model Representation

A fundamental problem for solid modelling in a virtual environment is model representation. In the graphics and VR community, active researches on model decimation, multi-resolution, level-of-detail management and zone culling are currently being carried out (Andujar et al. 2000; Gobbetti and Bouvier 2000; Kahler et al. 2001). Comparatively little research has been conducted for accommodating precise CAD models in a VR environment (Figueiredo and Teixeira 1994).

If CAD formats were directly used in a VR environment, the online processing time for visualizing a typical CAD model would make it impossible to interact in real time. The polygon model used in most VR systems provides the illusion of being immersed, but it may not be able to precisely define the object geometry. The use of a high-resolution model in a VR environment can increase model precision. The system may however not be able to respond in real time either. On the other hand, it is difficult to perform modelling because of the lack of topological relationships and constraint information in the polygon model. Therefore, it is necessary to develop a suitable model representation to support solid modelling in a VR environment. The model representation not only needs to support real-time visualization and interaction in a VR environment, but it also needs to support modelling activities as well as reflect the modelling process.
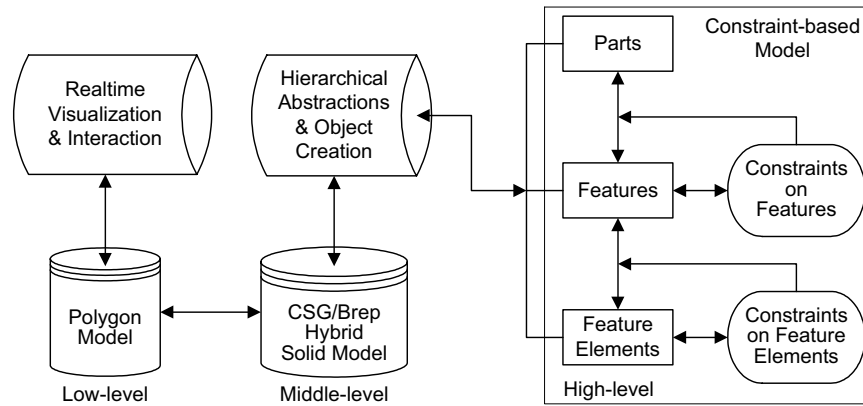
Fig. 2.1. Hierarchically structured and constraint-based data model

A hierarchically structured constraint-based data model is presented to support solid modelling in the VR environment (Figure 2.1). This data model includes five levels of information, i.e., parts, features, feature elements, geometric and topological relationships, and polygons. The definition of feature elements is the same as the feature entities in (Brunetti et al. 1995). The data model integrates a high-level constraint-based model for precise object definition, a mid-level CSG/Brep hybrid solid model for hierarchical geometry abstractions and object creation, and a low-level polygon model for real-time visualization and interaction in the VR environment. The information in the high-level model that is used for modelling can be divided into two types, i.e., object information on different levels and constraint information on different levels. An object can be a part, a feature or a feature element. The constraints on each object level that summarize the associativities between the individual objects on the same level not only provide precise object definition, but also provide a convenient way to realize precise 3D interactions. The mid-level solid model is the geometric and topological description of an object and is represented as a CSG/Brep hybrid structure. It not only provides the geometric and topological information of an object to support the hierarchical geometry abstractions and object creation, but also provides a convenient way for interactive feature-based solid modelling. The low-level polygon model provides the polygon data that corresponds to the mid-level Brep solid model for real-time visualization and interaction in the VR environment.

2.3.1 Structure of the Constraint-based Model

In the high-level constraint-based model, constraints are embedded in the solid model and organized at different levels to reflect the modelling process from features to parts (Figure 2.2).
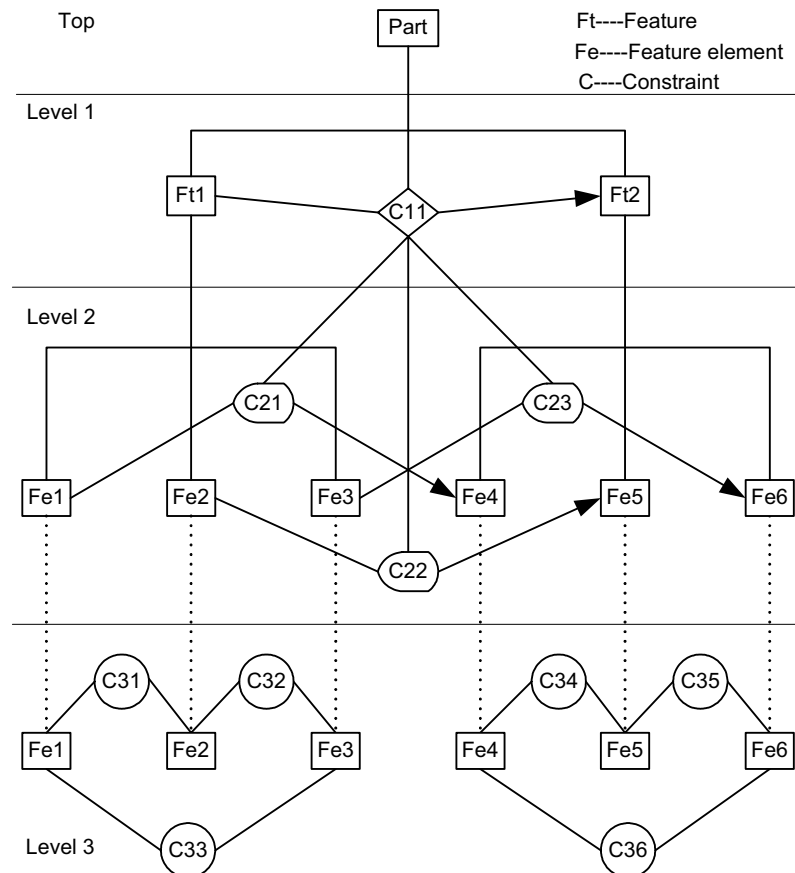
Fig. 2.2. The hierarchical structure of the high-level constraint-based model

Level 1 is the feature-based part model representation. Here, a part consists of features and the constraints between these features. The constraints on this level represent the spatial position relationships between the different features and they are called the external feature constraints. An external feature constraint has one direction and this direction is dependent on those of the external element constraints included in the external feature constraint.

Level 2 and Level 3 are the feature element based part model representation. The constraints on Level 2 and Level 3 are those between the feature elements. Since an external feature constraint between features is difficult to represent, a feature is sub-divided into a set of feature elements and the constraints between these feature elements. Correspondingly, an external feature constraint on Level 1 is decomposed into a set of constraints between the feature elements that individually belong to different features.
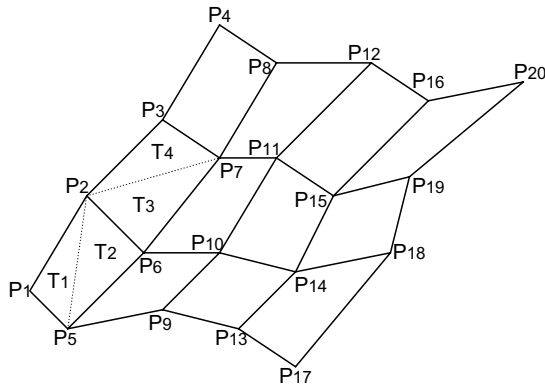
Fig. 2.3. Polygon model representation

The constraints on Level 2 represent the spatial position relationships between the feature elements that individually belong to different features, and they are called the external element constraints. An external feature constraint on Level 1 is sub-divided into a set of external element constraints on this level. An external element constraint has one direction and this direction points to the feature element that has been constrained. Typical external element constraints include against, alignment, distance, etc.

Level 3 provides the feature model representation. A feature consists of feature elements and the constraints between these feature elements. The constraints on this level represent the spatial position relationships between the feature elements belonging to a feature, and they are called the internal element constraints. The internal element constraints define the shape of a feature and are non-directional. They can be further divided into internal element geometric constraints and internal element topological constraints according to their properties. The internal elements geometric constraints represent the spatial position relationships between the feature elements that belong to a feature and are described as a face-based representation, such as parallel faces, perpendicular faces, distance faces and angular faces, etc. The internal element topological constraints represent the topological relationships between the feature elements that belong to a feature and are described as an edge-based representation, such as co-edge and co-circle.

2.3.2 Polygon Model Representation

The low-level polygon model is a triangle-based polygon representation that corresponds to the mid-level Brep solid model. The low-level polygon model describes each face in the Brep solid model as a two-level structure. On the bottom level locates the vertex array where the vertices that constitute a face are placed. On the top level locates the connect-lists that reflect the connecting

relationships of the vertices on the bottom level. Each connect-list separately defines a facet. It records the sequential numbers of the vertices that constitute a facet in the vertex array. In general, a face only has one vertex array and has some connect-lists that share the vertex array. For the face shown in Figure 2.3, the vertex array is (P_1, P_2, $\iiint P_{20}$) and the subscripts are the indices of the vertices in the array. The connect-lists are {1, 5, 2}, {2, 5, 6}, {2, 6, 7}, {2, 7, 3}, etc. Each facet is consisted of three vertices and is a triangle. The first triangle is $T_1 = \div P_1 P_5 P_2$, the second triangle is $T_2 = \div P_2 P_5 P_6$, the third triangle is $T_3 = \div P_2 P_6 P_7$ and the fourth triangle is $T_4 = \div P_2 P_7 P_3$, etc.

2.4 Constraint-based Manipulations

The framework of constraint-based manipulations is shown in Figure 2.4. For every object in the VR environment, such as a feature element, a feature, and a part, an event list, which is regarded as the attribute of this object, is attached to the object. An action list is connected to every event in the event list of an object. This action list shows the actions that will be performed as soon as the event occurs. Constraint-based manipulations are realized by these basic interactive events and actions will be performed when these events occur. The basic interactive events are attached to every object. Examples of basic interactive events are the grasping event, the moving event and the dropping event.

The grasping event has an action to acquire the current allowable motions of an object that it is attached to. An action for recognizing the constraints between individual objects is attached to the moving event and the dropping event. As soon as a user grasps an object, the grasping event occurs and the current allowable motions of this object are derived from the hierarchically structured constraint-based data model through constraints solving. Constraint-based manipulations are achieved by constraining the motions of 3D hands to the allowable motions. This is done by transferring the 3D motion data from the 3D input devices to the allowable motions of the object. The constraint-based manipulations not only ensure that the precise positions of an object can be obtained, but also guarantee that the existing constraints will not be violated in future operations.

Once a constraint has been recognized during the constraint recognition process, it will be highlighted and awaits a user's confirmation. Once it is confirmed, the recognized constraint will be satisfied precisely under the current allowable motions of the object and inserted into the hierarchically structured constraint-based data model. The satisfied constraint further restricts the subsequent motions of the object. While the constraint-based manipulations are being performed, the collision detection is switched on in order to detect possible collisions between the object being manipulated and other objects. If a collision is detected, the system will immediately provide a feedback to the user by highlighting the objects involved and the colliding sound.
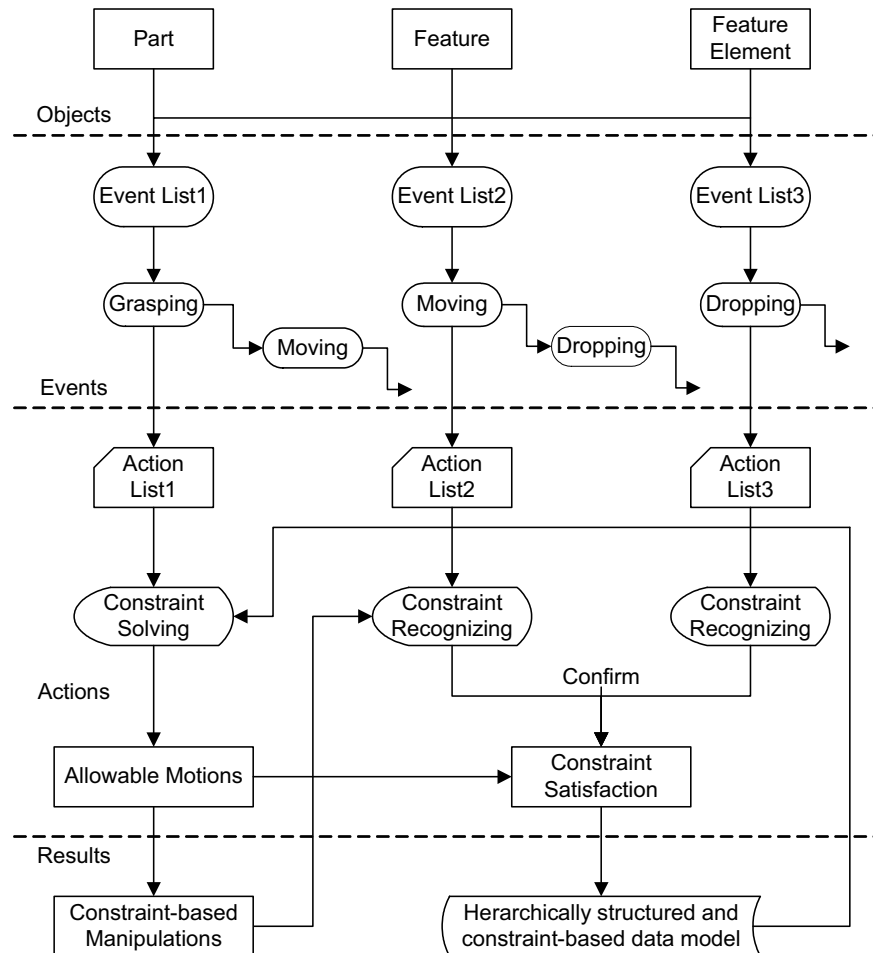
Fig. 2.4. The framework of constraint-based manipulations

2.4.1 Representation of Allowable Motions

The constraints between objects are implicitly created by the constraint-based manipulations with automatic constraint recognition and precise constraint satisfaction. Newly created constraints reduce the DOFs of the object being manipulated and implicitly confine future operations that can be applied to the object. The remaining DOFs define the allowable motions of the object. The allowable motions explicitly describe the next possible operations and ensure that future operations will not violate the existing constraints. The allowable motions

are represented as a mathematical matrix so that it can be conveniently derived from the constraints.

For every object in the free space, its configuration space has six DOFs: three translational DOFs and three rotational DOFs. To simplify the computation and clarify the presentation of the allowable motions, the configuration space is divided along three linear independent directions: X-axis, Y-axis and Z-axis. Therefore, some basic DOFs, i.e., three translational DOFs and three rotational DOFs can be obtained. Furthermore, the three basic translational or rotational DOFs are linearly independent of each other. Any remaining DOF that is used to define the allowable motions can be represented by these basic DOFs, therefore the allowable motions can be represented using these basic DOFs as the matrix in Eq. (2.1).

$$\begin{pmatrix} T_x & R_x & T_{x\min} & T_{x\max} & R_{x\min} & R_{x\max} \\ T_y & R_y & T_{y\min} & T_{y\max} & R_{y\min} & R_{y\max} \\ T_z & R_z & T_{z\min} & T_{z\max} & R_{z\min} & R_{z\max} \end{pmatrix} \tag{2.1}$$

In Eq. (2.1), the first column elements T_x, T_y and T_z are the linear translations along the X-axis, Y-axis and Z-axis respectively, and the second column elements R_x, R_y and R_z are the rotations about the corresponding axis respectively. The values of these elements in the matrix are either zero or one. Integer 1 indicates that motion is allowable in the direction along the corresponding axis. Integer 0 indicates that motion is not allowable in the corresponding axis direction. The third and fourth column elements are the allowable ranges of the three translations, which are defined by the minimum and maximum values of the three translations. For example, $T_{x\min}$ and $T_{x\max}$ are the minimum and maximum values of the translation along the X-axis. The fifth and sixth column elements are the allowable ranges of the three rotations, which are defined by the minimum and maximum values of the three rotations. For example, $R_{x\min}$ and $R_{x\max}$ are the minimum and maximum values of the rotation about the X-axis. If the translations or rotations along some axes are not allowable, the corresponding minimum and maximum values are zero.

2.4.2 Constraint Solving for Deriving Allowable Motions

Since most constraints are geometric constraints and they are shown as the limitations of the relative geometric displacements between objects, i.e., the limitations of the DOFs, the constraints applied to an object can be mapped to the DOFs of this object. In fact, the correspondence from constraints to DOFs can be extended to the correspondence from a set of constraints to the combination of DOFs. Therefore, the representation of constraints can be obtained by analyzing and reasoning the DOFs of an object, and constraints solving can also be regarded

as a process of analyzing and reasoning the DOFs of an object. Based on this, a procedure-based DOF combination approach is presented for solving 3D constraints (Figure 2.5). This approach has an intuitive manner for constraints solving since it combines DOF analysis with 3D direct manipulations in the VR environment.

As shown in Figure 2.5, the current allowable motions of an object are derived from the current remaining DOFs of the object. The action of grasping an object is interpreted by the constraint solver as requesting the current remaining DOFs of the object. The current constraints applied to the object can be obtained from the hierarchically structured and constraint-based data model. Initially, the object is unconstrained and has six remaining DOFs. If there is only one constraint applied to the object, the current remaining DOFs can be directly obtained by DOF analysis. If there are multiple constraints (more than one) applied to the object, the current remaining DOFs of the object can be obtained by DOF combination. The DOF combination for solving multiple constraints is based on the DOF analysis for solving individual constraints. Within the limitation of the current remaining DOFs determined by the current constraints, the object aims to satisfy a new constraint recognized by the current constraint-based manipulations applied to the object. The new constraint is precisely satisfied under the current allowable motions of the object and is further inserted into the hierarchically structured constraint-based data model to update the current constraints applied to the object. The update of the current constraints results in the update of the current remaining DOFs of the object and further results in the update of the current allowable motions of the object. Finally, the constraint-based manipulations applied to the object are updated correspondingly.
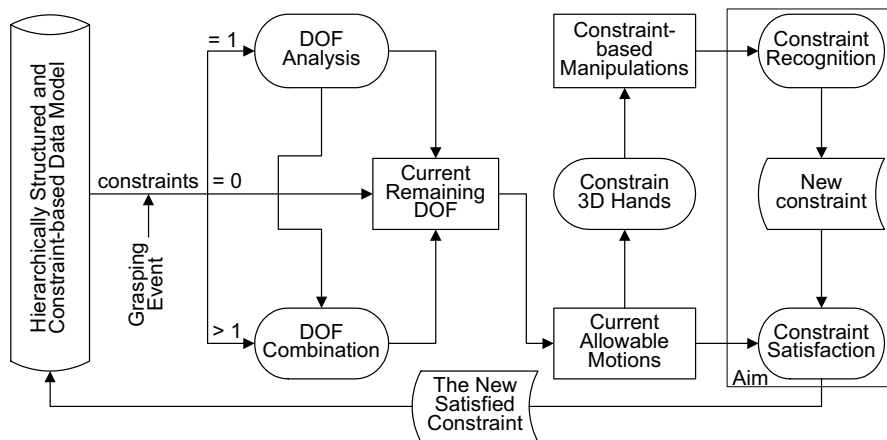

Fig. 2.5. The framework of procedure-based DOF combination constraint solving
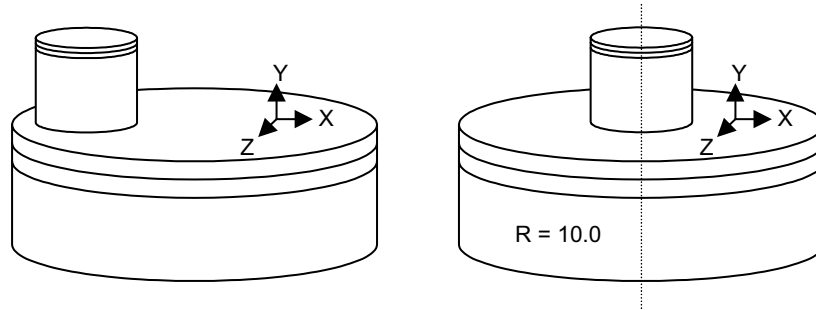
Fig. 2.6. The "against" and "line-alignment" constraints between two cylinders

DOF Analysis

Since DOFs are divided into three basic translational DOFs and three basic rotational DOFs, it is easy to connect a constraint with the remaining DOFs by analyzing the remaining basic translational and rotational DOFs corresponding to the constraint. On the other hand, the allowable motion matrix introduced in Section 2.4.1 is described by the three basic translational DOFs and the three basic rotational DOFs. Therefore, the allowable motion matrix corresponding to a constraint can be directly obtained by analyzing the remaining basic translational and rotational DOFs that correspond to the constraint. For example, if a small cylinder is placed on a big cylinder and they have to be axis-aligned (Figure 2.6), the constraints between the two cylinders are the "against" and "line-alignment" constraints. Using DOF analysis, the small cylinder has the translational DOFs T_x, T_z and the rotational DOF R_y for the "against" constraint, and the translational DOF T_y and the rotational DOF R_y for the "line-alignment" constraint. The allowable motion matrices that correspond to the two individual constraints are given in Eq. (2.2) and Eq. (2.3) respectively.

$$\begin{pmatrix} 1 & 0 & 410.0 & 10.0 & 0.0 & 0.0 \\ 0 & 1 & 0.0 & 0.0 & 0.0 & 2\phi \\ 1 & 0 & 410.0 & 10.0 & 0.0 & 0.0 \end{pmatrix} \tag{2.2}$$

$$\begin{pmatrix} 0 & 0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1 & 1 & 0.0 & 10.0 & 0.0 & 2\phi \\ 0 & 0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{2.3}$$

Similarly, the allowable motions matrices that correspond to other individual constraints can also be obtained by DOF analysis. Figure 2.7 gives the DOF analysis for some typical constraints.
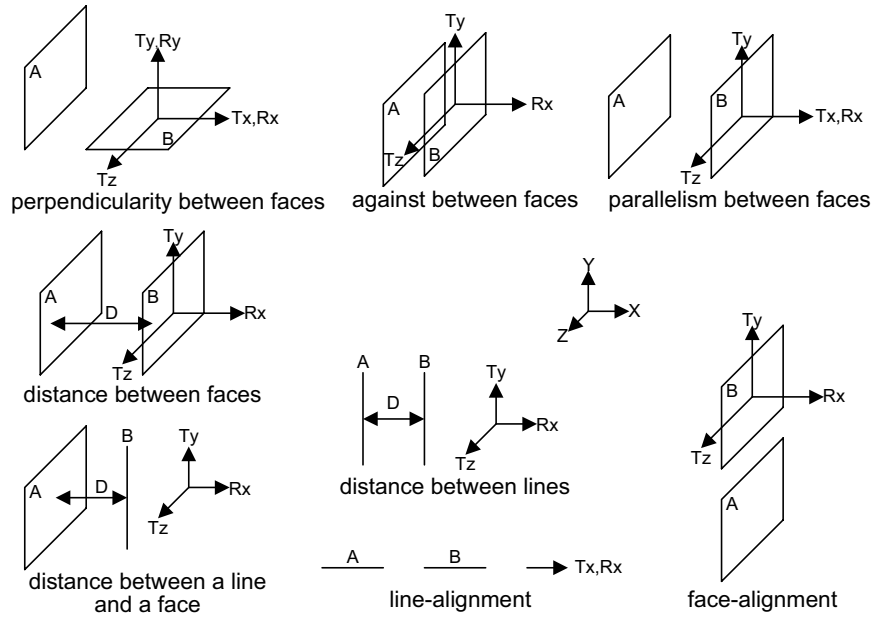
Fig. 2.7. DOF analysis to some typical constraints

DOF Combination

The DOF combination is used to represent the remaining DOFs that correspond to multiple constraints. It refers to the intersection of the DOFs of the allowable motions of the respective individual constraints.

An allowable motion is described by the three linearly independent translational DOFs and the three linearly independent rotational DOFs. Both the translational DOFs and rotational DOFs are a closed set respectively. Therefore, the DOF combination can be regarded as an individual combination of the six translational and rotational DOFs, and can be further represented as a combination of the allowable motion matrices of the respective individual constraints.

The combination of the allowable motion matrices can be realized using the "AND" Boolean operation of the allowable motion matrices of the respective individual constraints, i.e., the "AND" Boolean operations of the corresponding elements with the same position at the first and the second columns and the intersections of the allowable ranges of the translations or rotations along the same axis in the allowable motion matrices of the respective individual constraints.

In this way, the remaining DOFs of an object that correspond to the multiple constraints can be obtained and the allowable motion matrix that corresponds to these multiple constraints can also be acquired. For example, the allowable motion matrices that correspond to the "against" and "line-alignment" constraints in

Figure 2.6 are Eq. (2.2) and Eq. (2.3) respectively. Using DOF combination, the small cylinder has the rotational DOF R_y for the two constraints, and the allowable range of R_y is the intersection between the allowable ranges of R_y in Eq. (2.2) and Eq. (2.3), i.e., from 0.0 to 2ϕ. The final allowable motion matrix corresponding to the two constraints is shown in Eq. (2.4).

$$\begin{pmatrix} 0 & 0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0 & 1 & 0.0 & 0.0 & 0.0 & 2\phi \\ 0 & 0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix} \qquad (2.4)$$

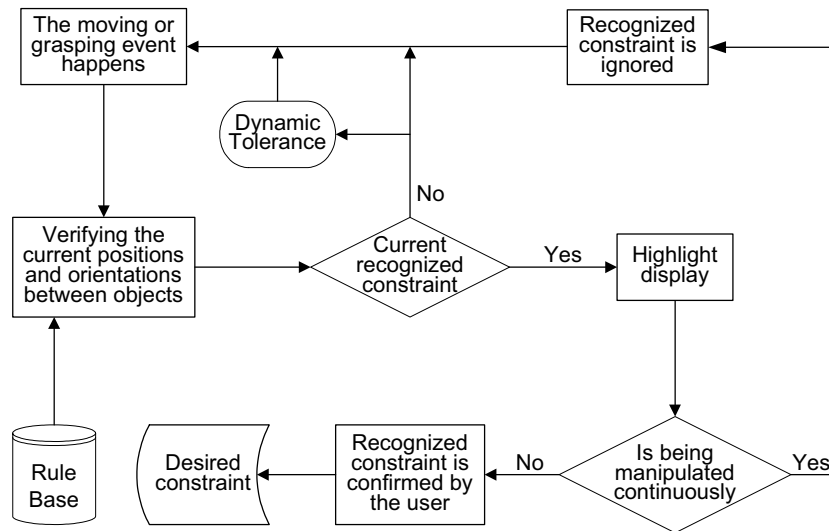### 2.4.3 A Rule-based Approach for Constraint Recognition

Fig. 2.8. Framework of rule-based constraint recognition

Constraints are implicitly incorporated into the VR environment for constraint-based manipulations through the automatic constraint recognition. Constraint recognition refers to the verification of the current positions and orientations between two objects to determine if they satisfy a particular type of constraint within a given tolerance. The constraint recognition framework is shown in Figure 2.8. While direct manipulations are being performed, as soon as a moving event or a release event occurs, an automatic constraint recognition process is triggered to detect all the possible constraints between the related objects. The system recognizes the constraints between objects from the current position and orientation of the manipulated object according to a rule base. The rule base defines some of the rules that are applied in constraint recognition for recognizing some specific constraints (Table 2.1). These constraints include against,

alignment, parallelism, perpendicularity, distance, co-circle, co-edge, etc. If the current positions and orientations between two objects satisfy the conditions of a constraint within a given tolerance, the matching constraint is recognized. Once a constraint is recognized within the given tolerance, it will be highlighted and awaits the user's confirmation. If the object is further manipulated continuously within a given time, the currently recognized constraint is ignored and the constraint recognition process is restarted. Otherwise, the currently recognized constraint is confirmed and the desired constraint is obtained. Furthermore, a dynamic tolerance is adopted in the constraint recognition process to improve the efficiency of this process. If the desired constraint is not recognized within the given tolerance, the tolerance is enlarged according to a given step until the desired constraint is recognized.

2.4.4 Some Special Constraint-based Manipulations

To reduce the search time for detecting the various types of constraints from various objects and enhance the modelling efficiency, some special constraint-based manipulations are also implemented as solid modelling operations in the VR environment. These operations include <placement>, <alignment>, <distance> and <insertion>. For each of these operations, the constraint recognition process is triggered to detect a particular pair of elements that satisfies some special constraint within a given tolerance.

The <placement> operation is responsible for locating one object relative to another object and is used as the initial locating operation of an object. It refers to an action of placing one object onto another object or placing two objects together. The constraint involved in this operation is an "against" constraint. If the recognized "against" constraint is precisely satisfied, the <placement> operation is stopped.

The <alignment> operation is responsible for locating one object relative to another object and is used as the precise locating operation of an object. The constraint involved in this operation is an "alignment" constraint. The <alignment> operation can be classified into two types according to the elements involved, i.e., <face-alignment> and <line-alignment>.

The <distance> operation is also responsible for locating one object relative to another object and is used as the precise locating operation of an object. The constraint involved in this operation is a "distance" constraint. The <distance> operation can be classified into three types according to the elements involved, i.e., <face-face distance>, <line-line distance> and <face-line distance>. The value of the distance is displayed near the object being manipulated for the user to acquire the precise distance. A toolbox with a cursor and displaying a number is also provided for the user to acquire the precise distance according to a given step.

The operations mentioned before are responsible for the precise location of an object before modelling and are called the locating operations. However, the <insertion> operation is used to perform a specific modelling task and is

responsible for determining the final position of an object. The basic motion of this operation is a translation. The constraint involved in the insertion operation is a "face-alignment" constraint.

Table 2.1. The rules for recognizing constraints

| |
|---|
| **Rules for detecting two against planar faces:** |
| ∉ Parallel: the product of the two unit normal vectors approaches to 0.0; and |
| ∉ Direction: the dot product of the two unit normal vectors approaches to -1.0; and |
| ∉ Close: the distance between a point on one facet to the projected point on the other facet is smaller than a given tolerance; and |
| ∉ Overlapping: the project of one facet on the other facet is not zero. |
| **Rules for detecting two aligning planar faces:** |
| ∉ Parallel: the product of the two unit normal vectors approaches to 0.0; and |
| ∉ Close: the distance between a point on one facet to the projected point on the other facet is smaller than a given tolerance. |
| **Rules for detecting two distance planar faces:** |
| ∉ Parallel: the product of the two unit normal vectors approaches to 0.0; and |
| ∉ Distance: calculate the distance between a point on one facet to the projected point on the other facet. |
| **Rules for detecting two paralleling lines:** |
| ∉ The product of the two unit vectors of the line segments approaches to 0.0. |
| **Rules for detecting two perpendicular lines:** |
| ∉ The dot product of the two unit vectors of the line segments approaches to 0.0. |
| **Rules for detecting two co-linear lines/axis:** |
| ∉ Parallel: the product of the two unit vectors of the line segments approaches to 0.0; and |
| ∉ Close: the distance between a point on one line to the projected point onto the other line is smaller than a given tolerance. |
| **Rules for detecting two distance lines/axis:** |
| ∉ Parallel: the dot product of the two unit vectors of the line segments approaches to 0.0; and |
| ∉ Distance: calculate the distance between a point on one line to the projected point onto the other line. |
| **Rules for detecting the face-linear distance:** |
| ∉ Parallel: the product between the unit normal vectors of the face and the unit vectors of the line segment approaches to 0.0; and |
| ∉ Distance: calculate the distance between a point on one line to the projected point onto the face. |
| **Rules for detecting two paralleling faces:** |
| ∉ The product between the unit normal vectors of the two faces approaches to 0.0. |
| **Rules for detecting two perpendicular faces:** |
| ∉ The dot product between the unit normal vectors of the two faces approaches to 0.0. |
| **Rules for detecting two co-edge faces:** |
| ∉ CE: all deviations between selected sample points on the two edges approach to 0.0. |
| **Rules for detecting two co-circle faces:** |
| ∉ CC: two circles with the same orientation and dimension. |

2.5 Implementation and Results
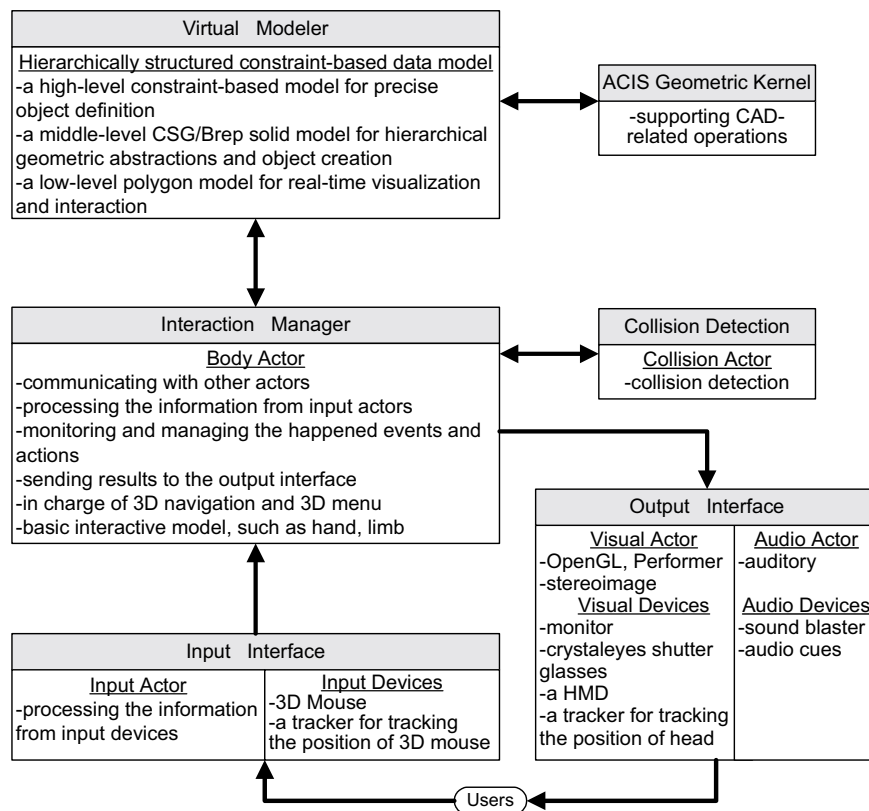
Fig. 2.9. System components

A prototype system for intuitive and precise solid modelling in a VR environment through constraint-based 3D direct manipulations has been implemented on the Division VR software based on a SGI Onyx2 with Infinite-Reality graphics workstation. The system components are shown in Figure 2.9. The body actor, that communicates with other actors, handles all aspects of user interaction. It receives and processes the information from the input actor, monitors and processes the events and actions in the VR environment and outputs the processed results to the visual actor and the audio actor. The collision actor resides in the system to detect the possible collisions between the objects in the VR environment. A 3D mouse controlled by the input actor is mainly used as the input device to carry out 3D manipulations. A six-degree-of-freedom head tracker, Head Mounted Display and CrystalEyes shutter glasses controlled by the visual actor are used for stereo display. Two sound blasters controlled by the audio actor

are used for audio. The VR modeller is in charge of all aspects of modelling to establish the hierarchically structured and constraint-based data model. It also provides the constraints to the body actor to generate constraint-based 3D direct manipulations. A geometric kernel ACIS is employed to support CAD-related operations.
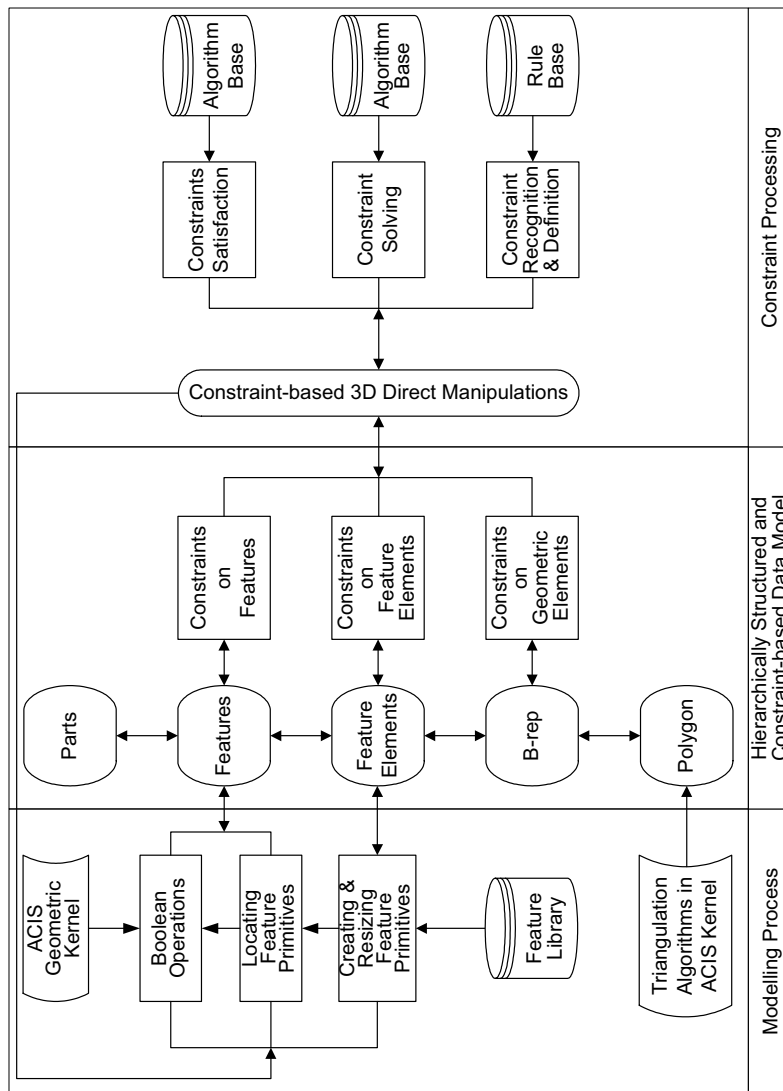

Fig. 2.10. System framework

The system framework is illustrated in Figure 2.10. It consists of three modules, i.e., the hierarchically structured and constraint-based data model, the

constraint processing and the modelling process. During the modelling process, parts are created from feature primitives using constraint-based manipulations through locating feature primitives and Boolean operations. A feature library is developed to provide some basic primitives to support solid modelling. The ACIS geometric kernel is employed to support Boolean operations, and polygon modelling through its triangulation function. The hierarchically structured constraint-based data model represents the entire solid modelling process with different design levels and the constraints at these levels. It also provides the constraints for generating precise constraint-based manipulations. In constraint processing, constraints are implicitly incorporated into the VR environment through constraint recognition and constraint satisfaction, and precise 3D constraint-based manipulations are generated from the constraints through constraint solving. The constraints in the hierarchically structured constraint-based data model are established through constraint-based manipulations during the modelling process. A rule base and related algorithm bases are developed to support constraint processing.
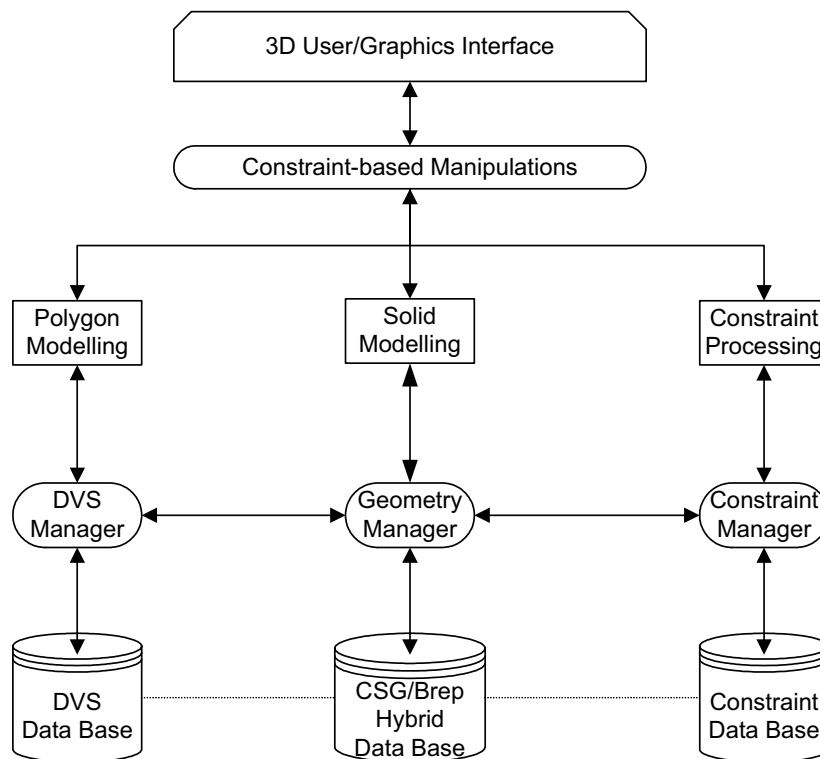

Fig. 2.11. Triple databases for solid modelling in the VR environment

For efficiently integrating CAD with VR, a triple database concept is adopted for VR-based solid modelling (Figure 2.11). Changes made in the DVS database are propagated into the CSG/B-rep hybrid database to change the geometry of the objects and further result in the variations in the constraint database to change or maintain the constraints between objects. On the other hand, changes made in the constraint database are propagated into the CSG/B-rep hybrid database to change the geometry of objects and further result in the variations in the DVS database. Changes made in the CSG/B-rep hybrid database are also propagated into both the DVS database and the constraint database to change the polygons of objects and change or maintain the constraints between objects.

2.5.1 Creation and Modification of Feature Primitives

Feature primitives, such as blocks, spheres, cylinders and cones, are regarded as the basic building blocks for solid modelling. A user can create a primitive through directly selecting the icon of a primitive in a 3D menu using a laser beam emitted from the hands. When the icon of a feature primitive is selected, the feature solid with standard sizes stands in the 3D space. At the same time, the corresponding feature elements and internal element constraints are instantiated and the corresponding Internal Feature Element Constraint Graph is established and stored in the hierarchically structured constraint-based data model. The user can directly resize the feature solid through constraint-based manipulations applied on the feature elements of the feature (the left image in Figure 2.12). The constraint-based manipulations are derived from the internal element geometric constraints applied to the feature element being manipulated. The actual dimensions of the feature are dynamically displayed near the manipulated elements for the user's reference. The user can also resize the feature solid by changing the parameters of the feature through a toolbox (the right image in Figure 2.12). During the resizing process, the feature shape is dynamically updated through solving the internal element topological constraints related to the feature element being manipulated.

2.5.2 Locating Feature Primitives

Feature primitives are located using some feature locating methods. The feature locating methods are provided by the combinations of the locating operations. The location of a feature primitive relative to other features in a part can be completely determined by each of the feature locating methods. Some typical feature locating methods are:
∉ «one <placement> operation + two <face-face distance> operations»
∉ «one <placement> operation + two <face-alignment> operations»
∉ «one <placement> operation + two <face-line distance> operations»

∉ «one <placement> operation + one <face-alignment> operation + one <face-face distance> operation»

∉ «one <placement> operation + one <face-line distance> operation + one <line-line distance> operation»

∉ «one <placement> operation + one <line-alignment> operation»

Two examples for locating feature primitives are shown in Figure 2.13. The left image in Figure 2.13 shows that the small block is located by the locating method «one <placement> operation + two <face-alignment> operations». The right image in Figure 2.13 shows that the cylinder is located by the locating method «one <placement> operation + one <line-alignment> operation».

2.5.3 Part Creation

The process of creating parts is to establish the external feature constraints between the features. In fact, the process of creating parts is also to establish the external element constraints since an external feature constraint is represented as a set of external element constraints.

Fig. 2.12. Resizing a feature primitive by constraint-based manipulations and a toolbox
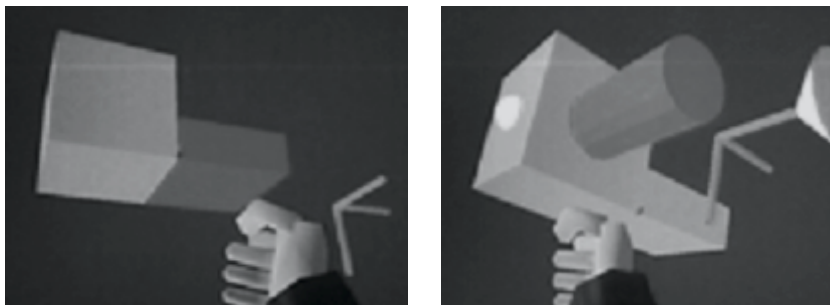

Fig. 2.13. Two examples for locating feature primitives

   Parts are created using Boolean operations after the feature primitives are located. The feature generated using the Boolean union operation directly inherits the constraints involved in the locating operations and these constraints are inserted into the hierarchically structured constraint-based data model. For the Boolean subtraction operation, the <insertion> operation is used to further determine the final position of the feature primitive before the Boolean subtraction. Since the feature primitive is located using the selected locating method, the <insertion> operation can be triggered after the constraint involved in the initial locating operation <placement> is released. It means that the constraint involved in the <insertion> operation replaces the constraint involved in the <placement> operation to further locate the feature primitive and the corresponding constraints are inherited by the feature generated from the Boolean subtraction operation. After Boolean operations, the system automatically checks whether there are any newly satisfied constraints between the generated feature and the reference features except for the directly inherited constraints. The newly satisfied constraints are also inserted into the hierarchically structured constraint-based data model. Figure 2.14 provides three parts that have been intuitively created using precise constraint-based 3D direct manipulations in the VR environment.
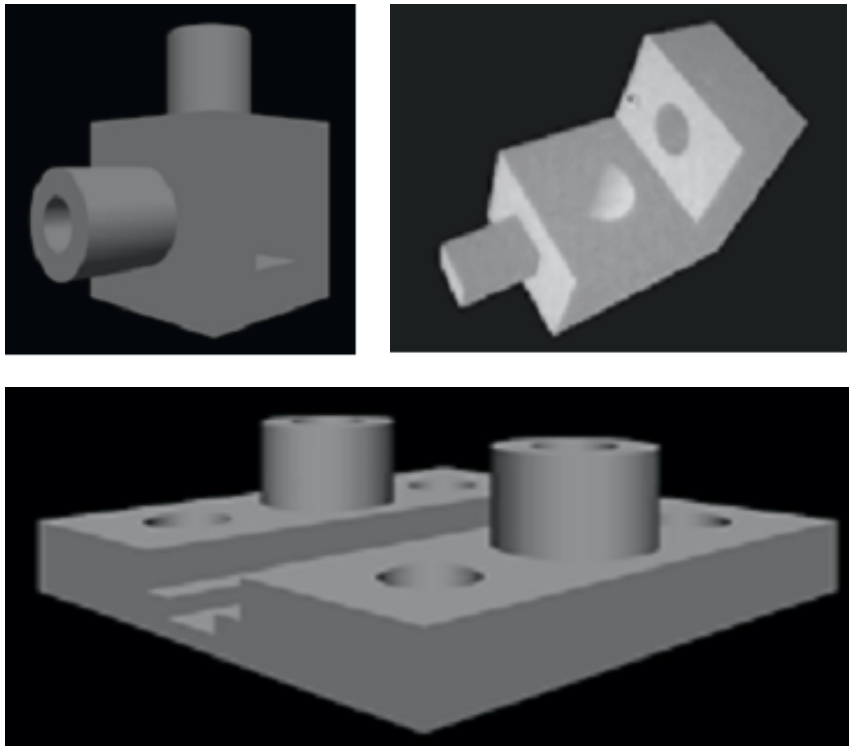

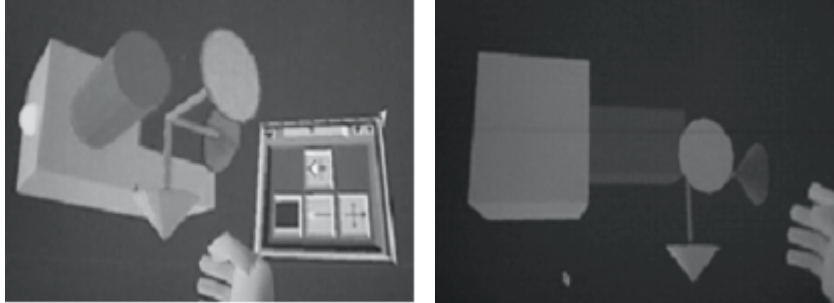**Fig. 2.14.** Three parts created in the VR environment

Fig. 2.15. Visual cues for constraint-based manipulations

2.5.4 Visual Cues for Constraint-based Manipulations

When performing solid modelling in the VR environment, visual cues are given to a user to obtain the desired constraint-based manipulations. This is done by visualizing the allowable motions of an object. The allowable motions can be clearly visualized to convene a message to a user for selecting desired constraint-based manipulations. As shown in Figure 2.15, a coordinate frame with a set of allowable motion flags and other visual cues are used to display the information of the allowable motions. The origin of the coordinate frame is located near the object being manipulated and the three axes represent the X-axis, Y-axis and Z-axis respectively. The normal arrow at the end of each axis indicates a translation DOF along the axis and the inverse arrow at the end of each axis indicates a rotational DOF about the axis.

2.6 Conclusions

With today's VR systems, it is difficult to directly and precisely create and modify objects in a VR environment. This chapter presents an approach for solid modelling in a VR environment. Solid modelling in the VR environment is performed precisely in an intuitive manner through constraint-based manipulations. A hierarchically structured and constraint-based data model is developed to support solid modelling in the VR environment. The data model integrates a high-level constraint-based model for precise object definition, a mid-level CSG/Brep hybrid solid model for hierarchical geometry abstractions and object creation, and a low-level polygon model for real-time visualization and interaction in the VR environment. Constraints are embedded in the solid model and organized at different levels to reflect the modelling process from features to parts. Constraint-based manipulations are accompanied by automatic constraint recognition and precise constraint satisfaction to establish the hierarchically

structured constraint-based data model, and realized by the allowable motions for precise 3D interactions in the VR environment. The allowable motions are represented as a mathematical matrix to conveniently derive the allowable motions from the constraints. A procedure-based DOF combination approach for 3D constraint solving is presented to derive the allowable motions. A rule-based constraint recognition engine is developed for both constraint-based manipulations and implicitly incorporating constraints into the VR environment. A prototype system has been implemented for precise solid modelling in an intuitive manner through constraint-based manipulations.

Continuous research is under active investigation, including model modification and assembly modelling in the VR environment. The refined approach is expected to be able to modify solid models through constraint-based manipulations and create assembly models from individual object models through constraint-based manipulations.

Acknowledgements

References

Aldefeld B (1988) Variation of geometries based on a geometric reasoning method. Computer Aided Design 20:117-126

Ambler AP, Popplestone RJ (1975) Inferring the position of bodies from specified spatial relationships. Artificial Intelligence 6:157-174

Andujar C, Saona-Vazquez C, Navazo I (2000) LOD visibility culling and occluder synthesis. Computer-Aided Design 32:773-783

Berta J (1999) Integrating VR and CAD. IEEE Computer Graphics and Applications 19:14-19

Bier EA (1986) Snap-Dragging. In: Proceedings of Computer Graphics, SIGGRAPH'86, vol 20, no 4, pp 233-240

Brunetti G, Martino TD, Falcidieno B, Habinger S (1995) A relational model for interactive manipulation of form features based on algebraic geometry. In: Proceedings of 3rd Symposium on Solid Modelling and Applications, Utah, USA, pp 95-104

Butterworth J, Davidson A, Hench S, Olano TM (1992) 3DM: a three-dimensional modeller using a head-mounted display. ACM Computer Graphics 25:197-208

Dani T, Gadh R (1997) COVIRDS: Shape modelling in a virtual reality environment. In: ASME 1997 Computers in Engineering Conference, Sacramento, CA, CD-ROM

Emmerik M (1990) A direct manipulation technique for specifying 3D object transformations with a 2D input device. Computer Graphics Forum 9:355-361

Fa M, Fernando T, Dew PW (1993) Interactive constraint-based solid modelling using allowable motion. In: Proceedings of 2nd ACM Symposium on Solid Modelling and Applications, Montreal, Canada, pp 243-252

Fernando T, Dew P, Fa M (1995) A shared virtual workspace for constraint-based solid modelling. In: Virtual Environment'95: Selected papers of the Eurographics workshops in Barcelona, Spain, Springer Wien, New York, pp 185-198

Fernando T, Muttay N, Tan K, Wimalaratne P (1999) Software architecture for a constraint-based virtual environment. In: Proceedings of the ACM Symposium on Virtual reality software and technology, London, UK, pp 147-154

Figueiredo M, Teixeira J (1994) Solid modelling as a framework in virtual environments. In: Rix J, Haas S, Teixeira J (eds) Proceedings of the IFIP WG 5.10 Workshops on Virtual Environments and Their Applications and Virtual Prototyping, Coimbra, Portugal, pp 99-112

Gao S, Wan H, Peng Q (2000) An approach to solid modelling in a semi-immersive virtual environment. Computer & Graphics 24:191-202

Gleicher M (1993) A graphics toolkit based on differential constraints. In: Proceedings of the ACM Symposium on User Interface Software and Technology, Atlanta, Georgia, pp 109-120

Gobbetti E, Bouvier E (2000) Time-critical multiresolution rendering of large complex models. Computer-Aided Design 32:785-803.

Gossard DC, Zuffante RP, Shakurai H (1988) Representing dimensions, tolerances and features in MCAE systems. IEEE Computer Graphics and Application 8(2):51-59

Hsu C, Alt G, Huang Z, Beier E, Bruderlin B (1997) A constraint-based manipulator toolset for editing 3D objects. In: Proceedings of Fourth Symposium on Solid Modelling and Applications, Atlanta, Georgia, pp 168-180

Kahler K, Rossl C, Schneider R, Vorsatz J, Seidel HP (2001) Efficient processing of large 3D meshes. In: Proceedings of International Conference on Shape Modelling and Applications, Genova, Italy, pp 228 –237

Khatib L (1996) Efficient interval constraint propagation with sequences. In: Proceedings of 2nd International Workshop on Constraint-Based Reasoning (CONSTRAINT-96), Key West, Florida, pp29-33

Kramer GA (1991) Using degrees of freedom analysis to solve geometric constraint systems. In: Proceedings of Symposium on Solid Modelling Foundation and CAD/CAM Applications, ACM Press, New York, pp 371-378

Kwaiter G, Gaildrat V, Caubet R (1997) Interactive constraint system for solid modelling objects. In: Proceedings of the ACM Fourth Symposium on Solid Modelling and Applications, Altanta, Georgia, pp 265-270

Lhomme O, Kuzo P, Mace P (1998) Desargues: A constraint-based system for 3D projective geometry. In: Bruderlin B, Roller D (eds) Geometric Constraint Solving and Applications. Springer-Verlag Berlin Heidelberg, pp 196-210

Liang J, Green M (1994) JDCAD: a highly interactive 3D modelling system. Computer & Graphics 18:499-506

Light R, Gossard D (1982) Modification of geometric models through variational geometry. Computer-Aided Design 14:209-214

Ma W, Tso SK, Zhong Y (1998) Constraint-based modelling in a virtual environment. In: Proceedings of CIRP Design Seminar on New Tools and Workflows for Product Development, Berlin, Germany, pp 221-232

Nishino H, Fushimi M, Utssumiya K (1999) A virtual environment for modelling 3D objects through spatial interaction. In: Proceedings of 1999 IEEE International Conference on Systems, Man, and Cybernetics, Oita University, Japan, pp 81-86

Sa AG, Zachmann G (1999) Virtual reality as a tool for verification of assembly and maintenance processes. Computer & Graphics 23:389-403

Stork A, Maidhof M (1997) Efficient and precise solid modelling using a 3D input device. In: Proceedings of Fourth Symposium on Solid Modelling and Applications, Atlanta, USA, pp 181-194

Sutherland IE (1963) Sketchpad: a man-machine graphical communication system. In: Proceedings AFIPS Spring Joint Computer Conference, Washington, pp 329-346

Whyte JN, Bouchlaghem AT, mcCaffer R (2000) From CAD to virtual reality: modelling approaches, data exchange and interactive 3D building design tools. Automation in Construction 10:43-55

Zhong Y, Yang H, Ma W (1999) A constraint-based approach for intuitive and precise solid modelling in a virtual reality environment. In: Proceedings of the Sixth International Conference on Computer Aided Design & Computer Graphics, Shanghai, China, pp 1164-1171