Interactive Engineering Assembly within a Virtual Environment

M. Munlin
Information Technology Department
Sirindhorn International Institute of Technology (SIIT)
Thammasat University, P.O. Box 22
Pathumthani 12121, Thailand
Tel : (662) 9869101-8 ext 2003 FAX: (662) 986-9113
Email: amin@siit.tu.ac.th

Abstract

This paper presents interactive constraint-based 3D manipulation techniques that can be used within a virtual environment to design and assemble solid models purely by 3D manipulations. An *automatic constraint recognition* technique is used to recognize assembly relationships and geometric constraints such as *against*, *coincidence*, *tangency* and *concentricity* between solid models in the virtual environment. The constraints recognized from 3D manipulation are maintained in a relationship graph.

A technique referred to as *allowable motion* is used to automatically constrain the subsequent 3D manipulation of the solid model without invalidating its associated constraints. This technique supports the accurate 3D positioning in terms of geometric constraints.

A Graph-based algorithm has been developed to support the propagation of 3D manipulations among constrained solid models in a realistic manner.

A prototype system has been implemented to demonstrate the feasibility of these techniques for model construction and assembly operations within a virtual environment. Several industrial case studies such as a puma robot and a shaft mounted speed reducer have been successfully demonstrated.

Keywords: 3D Interaction Technique, Constraint-Based Solid Modelling, Geometric Modelling, Graphs Algorithms, CAD/CAM.

1. Introduction

The recent advent of Virtual Reality (VR) technology has made it feasible to directly interact with objects in 3D space. This offers the potential to develop highly interactive 3D user interfaces for a variety of engineering applications such as tele-operation, data visualization, virtual manufacturing, collaborative working, assembly modelling and geometric modelling [1,2]. A number of commercial and non-commercial VR environments are currently available to support the development of virtual environments for such applications. However, a common weakness of the existing virtual environments is the lack of efficient geometric constraint management facilities such as run-time constraint detection and the maintenance of constraint consistencies for 3D manipulations. As noted by several researchers [3,4], this lack of support for constraints makes it difficult to achieve the accurate 3D positioning of solid models using 3D input devices, and hence 3D solid modelling operations are impractical in the current virtual environments.

The authors have conducted research to develop techniques for supporting geometric constraint management facilities within virtual environments. As a result, two 3D interaction techniques called *allowable motion* and *automatic constraint recognition* have been developed. The automatic constraint recognition technique is capable of recognizing assembly relationships and geometric constraints between solid models from the user's 3D manipulations. The constraints are stored in a directed graph called a *Relationship Graph* (RG). The allowable motion technique computes the remaining degrees of freedom of a solid model from its constraint information in the RG. It is also used to provide an efficient means of satisfying 3D constraints simply by applying a number of transformation

RECEIVED 25 JANUARY, 2001
ACCEPTED 7 AUGUST, 2001

operations to the solid model. In comparison with other research work [5,6,7], these two techniques provide a more intuitive way of specifying 3D constraints by 3D manipulations and support a larger set of constraints such as *parallel symmetry, angular symmetry, parallel, coincidence, tangency, concentricity, perpendicular, cylindrical fit against, spherical fit, rack-pinion contact,* and *gear contact.* Additionally, allowable motion has been exploited for automatically constraining 3D manipulations of a solid model without invalidating its associated constraints. This has resulted in an intuitive means of achieving the accurate 3D positioning of the solid model by a 3D input device, without the need for many menu interactions as in previous systems [5,8].

The automatic constraint recognition and allowable motion techniques have been used to develop an interactive constraint-based solid modelling kernel [9]. This kernel has been integrated with a virtual environment to support solid modelling and assembly modelling operations through intuitive 3D graphical interaction. The paradigm presented to the user, within this virtual environment, is the design of solid models by manipulating a set *of virtual design tools* [9].

In previous work [5,6,7], auto-constraint techniques have been developed that infer constraints from 2D dragging operations or from transformation operations. Unfortunately, these techniques are restricted to 2D interaction and support a limited number of constraint types. The automatic constraint recognition technique reported in this paper is able to recognize 3D constraints between geometric entities from the user's 3D manipulations. This technique provides a more intuitive way of specifying 3D constraints than existing techniques. The technique is applicable to both constraint-based solid modelling and assembly modelling in a uniform manner.

The graph-based algorithm is used to simulate the kinematics behavior of assemblies in respond to the user's 3D manipulations in a realistic manner. New constraints can be recognized and satisfied during the 3D manipulations and the kinematics behavior of assemblies can be automatically updated. This avoids the need to pre-define a complete set of constraint information for an assembly model as in many commercial systems [10].

2. Integrating 3D Interaction and Constraints

This section discusses the relationship graph, automatic constraint recognition and allowable motion techniques. These techniques have integrated 3D interaction and constraints to support the efficient management of geometric constraints within a virtual environment [9,11]. Before describing these techniques, it is necessary to clarify the following terms used throughout the remaining part of this paper.

- *Tar-SM* refers to the solid model that is being manipulated by the user, and *Ref-SM* set refers to all the solid models in the 3D world, except for the *Tar-SM.*
- Object refers to a solid model.
- A solid model is comprised of geometric elements.
- *Tar-GE:* target geometric element of the *Tar-SM.*
- *Ref-GE:* reference geometric element of the *Ref-SM.*
- *Geometric Elements* (GEs) can be one of the following types: *point, line, circle,* and *halfspaces.*
- *Halfspace* refers to a surface that divides the 3D space into two regions: solid and non-solid. These halfspaces are combined through Boolean set-theoretic operations to create primitives (e.g. block and cylinder), that in turn are combined to create complex solids. This solid modelling technique is referred to as *Constructive Solid Geometry* (CSG) [12].
- *Geometric Entity* refers to either a solid model or a geometric element.
- *Constraint* refers to either an assembly relationship or a geometric constraint.
- *Assembly Relationship* refers to a mating condition between assembly components such as *against, cylindrical fit,* and *gear contact.*

• *Geometric Constraint* refers to a relationship between geometric entities such as *distance, coincidence, tangency, angle, parallel*, and *perpendicular*.

2.1 Relationship Graph (RG)

The RG maintains geometric constraints or assembly relationships between GEs. The RG is a directed graph where each node represents either a geometric element or an assembly component. Every arc of the RG corresponds to constraints between two GEs. The direction of the arc denotes the constraint dependency between two GEs. The GE that the arc points to, depends on the GE that the arc departs from. This implies that the changes in the geometry of the latter are always propagated to the former. In this directed graph, all the constraints associated with a GE are classified into two groups: *in* and *out*. The *in* constraints are those constraints that are imposed on the GE to specify its geometry (i.e. positions and sizes). The *out* constraints are those constraints that are propagated by the GE to its dependent GEs.

The RG described above is not a solid representation scheme but it is concerned with only representing constraints. These constraints are evaluated to determine the geometric parameters of each geometric element and the 3D location of each component in an assembly. A CSG graph [12] is used for representing the geometry of assembly components in a hierarchical form. The links are maintained between the nodes of the RG and its corresponding nodes of the CSG graph. The geometric definition of the boundary face of each halfspace is also maintained in a separate data structure.

2.2 Automatic Constraint Recognition

The automatic constraint recognition process carries out the construction of the RG. While the *Tar-SM* is being manipulated using a 3D input device, a motion event handler continuously samples the input to generate a sequence of 3D motion events (e.g. 3D translation and rotation). At every sampled point, the constraint recognition process is triggered to identify any possible constraints between the geometric elements of the *Tar-SM* and geometric elements of the *Ref-SM* set. A constraint is recognized if two geometric elements satisfy the conditions for a particular constraint type within a given tolerance. Constraints such as *angular symmetry, against, coincidence, tangency, concentricity, parallel, perpendicular, gear contact* and *cylindrical fit* are automatically recognized by this process. Two examples are given in Figure 1 to illustrate this process.

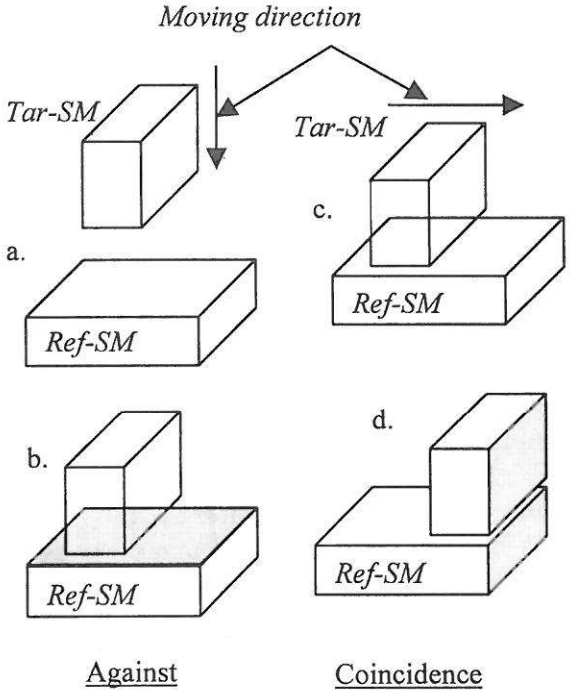


Figure 1. Constraint Recognition Process

When a new constraint is recognized, a visual feedback is provided to the user by highlighting the relevant geometric elements (e.g. faces, edges and vertices). If the user continues to move the *Tar-SM*, the newly identified constraint is ignored and the constraint recognition process is continued. On the other hand, if the user does not move the *Tar-SM* within a prescribed time, the newly identified constraint is inserted into the RG by attaching the constraint to the *in* constraint list of the *Tar-GE* and *out* constraint list of the *Ref-GE*. This constraint is also inserted into the *in* constraint list of the *Tar-SM*, and *out* constraint list of the *Ref-SM*. Once a constraint is inserted,

the allowable motion of the *Tar-SM* is updated as explained in the next section.

2.3 Allowable Motion

The six degrees of freedom provided by a 3D input device should be constrained in some way to help the user to accurately move an object to a desired 3D position [3,4]. In our approach, an *allowable motion* is used as a means of automatically constraining 3D manipulations of a solid model, thereby achieving the accurate 3D positioning of the solid model purely through graphical 3D interaction. This technique is also used to provide an efficient means of satisfying constraints by applying a number of transformation operations to the geometric entity.

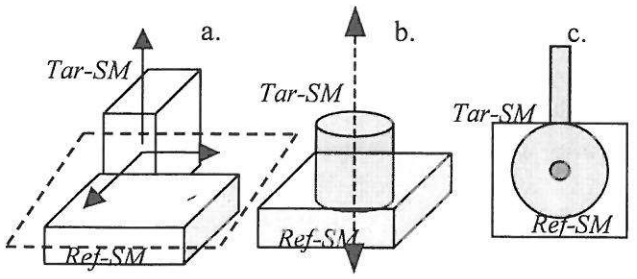
2.3.1 Definition of the Allowable Motion

The term allowable motion refers to the remaining degrees of freedom for an under-constrained solid model or geometric element. An allowable motion is defined by two elements:

(i) *Motion Type* which refers to either allowable translations or rotations; and

(ii) *Motion Geometric Element* (Motion-GE) on which the origin of the local coordinate system (LCS) of the solid model can be translated, or about which the LCS can be rotated.

The *Motion-GE* can be a point, line or plane. Five types of allowable motion can be identified depending on the motion type and *Motion-GE*. These allowable motion types are explained below with examples in Figure 2.



Against:	Cylindrical fit:	Spherical fit:
TR_PL	*TR_LN*	*NO-TR*
RT_LN	*RT_LN*	*RT_PT*

Figure 2. Examples of Allowable Motion

2.3.2 Allowable Translations

Four types of allowable translations are identified.

1. **Translation on Plane** (*TR-PL*): The origin of the LCS of the *Tar-SM* can be translated on a plane. An example for this case is shown in Figure 2a, where against constraint is imposed on the *Tar-SM*. With this constraint, the translation of the *Tar-SM* on the top plane (*Ref-GE*) of the *Ref-SM* can be considered as *TR-PL* type. In this case, the *Motion-GE* is a plane that is parallel to the *Ref-GE* plane and passes through the origin of the LCS of the *Tar-SM*.

2. **Translation on Line** (*TR-LN*): The origin of the LCS of the *Tar-SM* can be translated along a line. An example for this case is shown in Figure 2b, where a line element (*Tar-GE*) of the *Tar-SM* is concentric with the cylindrical hole (*Ref-GE*) of the *Ref-SM*. In this example, the translation of the *Tar-SM* along the *Tar-GE* line can be considered as *TR-LN* type. In this case, the *Motion-GE* is a line which is parallel to the *Tar-GE* line and which passes through the origin of the LCS of the *Tar-SM* (the axis of cylinder).

3. **Translation in 3D Space** (*TR-3D*): The origin of the LCS of the solid model can be translated in 3D space. For example, if only a *parallel* constraint is imposed on the *Tar-GE* plane of the *Tar-SM*, the *Tar-SM* can be translated in 3D space while maintaining the *parallel* constraint as valid.

4. **Non Translation** (*NO-TR*):The origin of the LCS is fixed or can not be translated.

2.3.3 Allowable Rotations

Three types of allowable rotations are identified.

1. **Rotation about Point** (*RT-PT*): The LCS of the solid model can be rotated about a point. An example for this case is shown in Figure 2c, in which a vertex (*Tar-GE*) of the *Tar-SM* is coincident with a vertex of the *Ref-SM*. The rotation of the *Tar-SM* about the *Tar-GE* point can be considered as *RT-PT* type. In this case, the *Tar-GE* point is treated as the *Motion-GE*.

2. **Rotation about Line** *(RT-LN)*: The LCS of the solid model can be rotated about a line. Figure 2a and Figure 2b can be used as examples for this case. In Figure 2a, the rotation of the *Tar-SM* about the normal vector of the *Tar-GE* plane can be treated as *RT-LN* type. In this case, the *Motion-GE* (rotation axis) is parallel to the normal vector of the *Tar-GE* plane and passes through the origin of the LCS of the *Tar-SM*. Similarly, in Figure 2b, the rotation of the *Tar-SM* about the *Tar-GE* line can be considered as *RT-LN* type. In this case, the *Tar-GE* line (axis of the *Tar-SM)* is treated as the *Motion-GE*.

3. **Non Rotation** *(NO-RT)*: The origin of the LCS is fixed or can not be rotated.

2.3.4 Derivation of Allowable Motion for Several Constraints

When several constraints are associated with the *Tar-SM*, the resulting allowable motion is found by intersecting the allowable translations and rotations imposed by each single constraint. This is carried out for the *Tar-SM* and the *Tar-GE*s separately. The intersection of two allowable translations is a translation on the intersection geometry of their *Motion-GE*s. Similarly, checking the relationship between their Motion-GEs derives the intersection of two allowable rotations [9,11]. An example is explained below to show how to derive the allowable translation and rotation from two constraints (*against* in Figure 3a and *coincident* in Figure 3b) for the *Tar-SM*.

1. **Intersecting allowable translation of the *Tar-SM***: For the two allowable translations of *TR-PL* type, the intersection geometry of their *Motion-GE*s is a line (labeled *New TR-LN* in Figure 3b). Therefore, the resulting allowable translation of the *Tar-SM* is *TR-LN* along the intersection line.

2. **Intersecting allowable rotation of the Tar-SM**: For the two allowable rotations of *RT-LN* type, their *Motion-GE*s (labeled *RT-LN1* and *RT-LN2*) are not coincident, or not parallel. Therefore, the resulting allowable rotation of the *Tar-SM* is *NO-RT*.

Table 1 and 2 summarize the results for the intersection of allowable translations and rotations respectively.

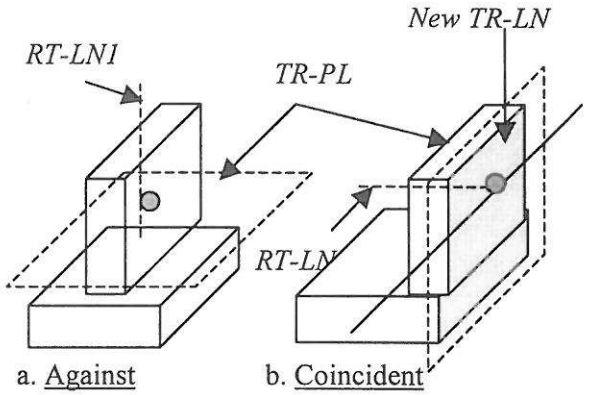


Figure 3. Deriving Allowable Motions for Several Constraints

TR-Type	TR-3D	TR-PL	TR-LN
TR-3D	*TR-3D*	*TR-PL*	*TR-LN*
TR-PL	*TR-PL*	*TR-LN*	*NO-TR*
TR-LN	*TR-LN*	*NO-TR*	*NO-TR*

Table 1. Intersection of allowable translations

RT-Type	RT-PT2		RT-LN2	
RT-PT1	coincide	intersect	coincide	intersect
	RT-PT1	*RT-LN3*	*RT-LN2*	*NO-RT*
RT-LN1	coincide	Intersect	coincide	Intersect
	RT-LN1	*NO-RT*	*RT-LN1*	*NO-RT*

Note: RT-LN3 is a line that passes through both PT1 and PT2

Table 2. Intersection of allowable rotations

2.4 Constraint Satisfaction using Allowable Motions

When a constraint is recognized within a given tolerance from 3D manipulations, and inserted into the RG, this constraint needs to be accurately satisfied from the *Ref-SM* to the *Tar-SM*. This is achieved by applying a number of transformation operations to the *Tar-SM*. These transformation operations are confined to the allowable motions of the *Tar-SM* without invalidating its previously satisfied constraints. Once the constraint is satisfied, the allowable motions of the *Tar-SM* are re-computed, thus further restricting the subsequent actions involved in satisfying new constraints. Figure 3

is used to illustrate this further. In this example, the *Tar-SM* is initially constrained relative to the *Ref-SM* in terms of *against* constraint (Figure 3a). Its allowable motions are *TR-PL* and *RT-LN*. In Figure 3b, a *coincidence* constraint between two planes is to be imposed on the *Tar-SM*. To satisfy this constraint, a rotation operation is first derived which rotates the *Tar-SM* about the *Motion-GE* of the *RT-LN1* (in Figure 3a). This operation enables the *Ref-GE* and *Tar-GE* to be parallel. Secondly, a translation operation is performed that translates the *Tar-SM* on the *Motion-GE* of the *TR-PL* by a distance in order to make the *Ref-GE* and *Tar-GE* exactly coincident as shown in Figure 3b. The allowable motions of the *Tar-SM* are then updated. In this case, the resulting allowable motions are *TR-LN* and *NO-RT*.

3. Graph-based Algorithms for Constraint Satisfaction

As explained in the previous section, the allowable motion of the *Tar-SM* is derived relative to its *Ref-SM*s and is then used to satisfy the constrains. However, this is not sufficient for interactively simulating the behaviors of constrained models. For example, by simply constraining the 3D manipulations of the *Tar-SM* against its own allowable motions, it is not possible to allow the user to drag the lower-arm of the puma robot to reach an arbitrary 3D position.

This section describes graph-based algorithms that address the above problem. The basic idea is to convert 3D manipulation into a constraint satisfaction process, by consuming the allowable motions of solid models in the RG. Such constraint satisfaction is performed in an incremental manner without re-satisfying all the constraints associated with objects from scratch. The incremental constraint satisfaction is achieved by exploiting the constraint dependencies in the RG. As a result, an interactive response can be achieved for 3D manipulations.

3.1 Constraint satisfaction within a directed acyclic graph

The term *Directed Acyclic Graph* (DAG) refers to the RG that does not contain loops. A typical example for a DAG is given in Figure 6a.

Figure 6 is a simplified "stick-figure" of the puma robot in Figure 5. In this example, four components of the robot are considered for this demonstration: *Base* (B), *Shoulder* (S), *Upper-arm* (U) and *Lower-arm* (L). These four components are assembled together through the following sequence. The *Shoulder* is placed on the top of the *Base* by recognizing an *against (A)* constraint. These two components are then aligned along their axes through a *concentricity (C)* constraint. The *Upper-arm* is assembled with the *Shoulder* by recognizing a *cylindrical fit (F)* and *against* constraint. In a similar way, the *Lower-arm* is assembled with the *Upper-arm* (Figure 4 and Figure 5).

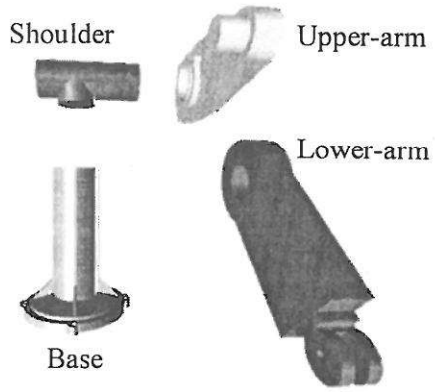


Figure 4. Initial position of a puma robot

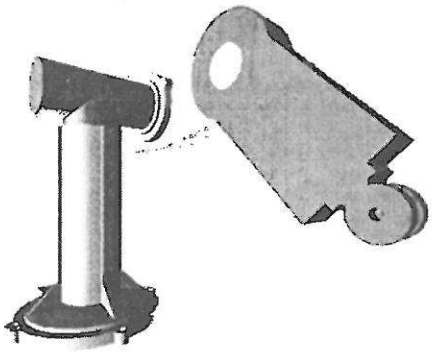


Figure 5. Final position of a puma robot

Following the above sequence, the RG in Figure 6a is created to maintain the constraints among the components. Other forms of the RG may be created if the user assembles four components through other sequences. However, the resulting RG is always a directed acyclic graph. It should be noted that in Figure 6a, a single arc between two nodes may correspond to more than one constraint between two components. The allowable motions are derived and associated with each node to represent the remaining degrees of freedom of each component with respect to its parent component. In this example, the allowable motion of each component except for the *Base* is *RT-LN*.

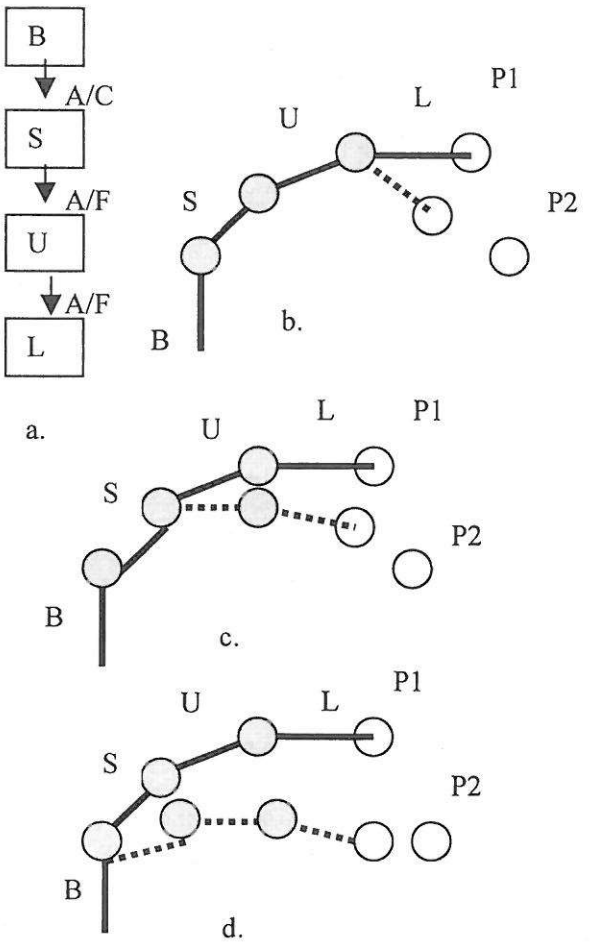


Figure 6. Constraint Satisfaction in a Directed Acyclic Graph

The 3D manipulation of the *Tar-SM* is handled as follows. First, 3D manipulation received from the 3D input device is converted into a constraint satisfaction process. For example, a 3D translation can be considered as a constraint satisfaction process to satisfy a coincidence constraint between a point on the *Tar-SM* and a target point. A 3D rotation can be treated as a constraint satisfaction process to satisfy an angle constraint between an axis of the local coordinate system of the *Tar-SM* and a target line. This constraint is then satisfied by consuming the allowable motions of the *Tar-SM* and its up-stream nodes in the RG. This is carried out in a recursive manner, until the constraint is satisfied. However, it is possible that a 3D constraint created for 3D manipulation may not be fully satisfied. In this case, iteration terminates when the best possible solution is found. During constraint satisfaction, any changes in a node are propagated to its down-stream children nodes through graph traversal.

For example, suppose the user is dragging the point P1 on the *Lower-arm* towards P2. First, a coincidence constraint between P1 and P2 is inserted into the RG. The *Lower-arm* is then rotated in terms of its allowable motion, attempting to satisfy the constraint (Figure 6b). Since this coincidence constraint can not be fully achieved by the rotation of the *Lower-arm*, the allowable motion of its parent node *(Upper-arm)* is used to move the point P1 further towards P2 (Figure 6c). Similarly, the allowable motion of the *Shoulder* is also consumed to satisfy the constraint (Figure 4d). Finally, since the *Base* is not constrained, the *Base* can be moved in such a way to enable P1 and P2 to be coincident. However, if the *Base* is constrained, recursive constraint satisfaction may be necessary.

3.2 Constraint satisfaction within a rigid graph

This refers to the case where the *Tar-SM* is fully constrained relative to one parent only, and their corresponding RG nodes are therefore combined into a *rigid node*. The RG node of the *Tar-SM* is marked as a rigid node while its parent node is marked as the root node of the rigid node. In addition, the attribute for each arc between these two nodes is marked to distinguish these links from other links

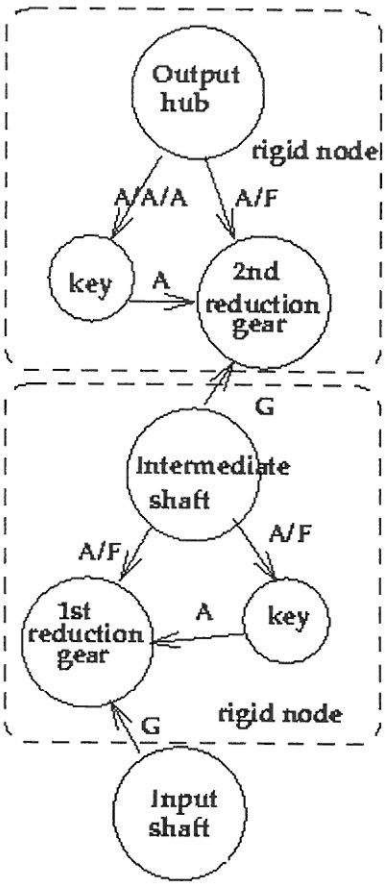
departing from the root node. By rigid graph re-writing, subsequent constraints which are imposed on any nodes inside the rigid node will be re-directed to the root of the rigid node. This means transforming all the objects of the rigid node, as a whole will satisfy these constraints. Figure 7 illustrates the rigid graph re-writing of a shaft mounted speed reducer. In this figure, one key is used to fix the second reduction gear with the output hub sub-assembly while another key is used to fix the first reduction gear with the intermediate shaft. The rigid node for the output hub, the key and the second reduction gear, as shown in the dashed box of Figure 7 is created as described below. The key is first fully constrained inside the keyway of the output hub in terms of three *against* constraints. Therefore, the rigid node is created by combining the RG nodes of the key and the output hub into a single node. The second reduction gear is fully constrained in terms of three constraints: *cylindrical fit* and *against* with the output hub, and another *against* with the top face of the key. Since the second reduction gear is fully constrained relative to one node only (i.e. rigid node), its RG node is then combined into this rigid node. The rigid node for the intermediate shaft, the first reduction gear and the key is created in a similar manner.

Once the rigid node is created, any changes in the transformation of the rigid node will be propagated to all components of a rigid node. For example, when the user rotates the input shaft. The pinion of the input shaft drives the first reduction gear that in turn drives the intermediate shaft whose pinion drives the second reduction gear. The second reduction gear drives the output hub that can be attached to the drive shaft of a system.

4. Implementation and Results

A prototype system has been developed on SGI workstations such as Indigo, Indy, and Indigo2. The IRIS Inventor supports the visualization of the virtual environment.

The interactive assembly process has been tested against the shaft mounted speed reducer (gearbox) to demonstrate the capability of the system. The system enables the user to carry out assembly operations simply by 3D manipulations. In particular, the system is capable of automatically recognizing assembly



A: against, F: cylindrical fit, G: gear contact
Figure 7. RG of the shaft mounted speed reducer illustrates a rigid node

relationships such as *against, cylindrical fit,* and *gear contact* purely from user's 3D manipulation. The allowable motion is used for satisfying these assembly relationships.

The constraint recognition (*against, cylindrical fits*, and *gear contact*), allowable motion *(RT-LN, TR-LN, TR-PL, TR-3D,* and *NO-RT*) and graph-based algorithms for propagating 3D manipulations in DAGs and rigid graph have been successfully demonstrated. Figure 8 – 15 demonstrate how the shaft mounted speed reducer can be assembled using the 3D manipulation techniques discussed in this paper. Once the model is assembled, its kinematics behavior can be simulated in response to the user's 3D

manipulation using the constraint satisfaction and propagation techniques as described in Section 3.

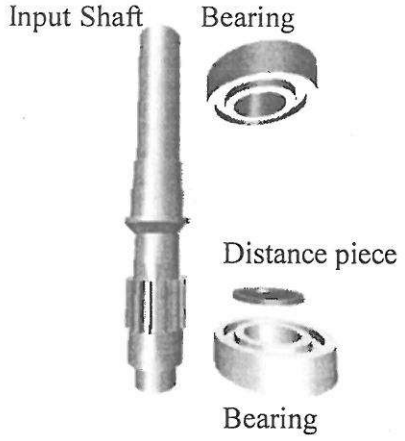


Figure 8: Initial input shaft sub-assembly that will be assembled by 3D manipulations.

Figure 9: Final input shaft sub-assembly that is assembled by 3D manipulations.

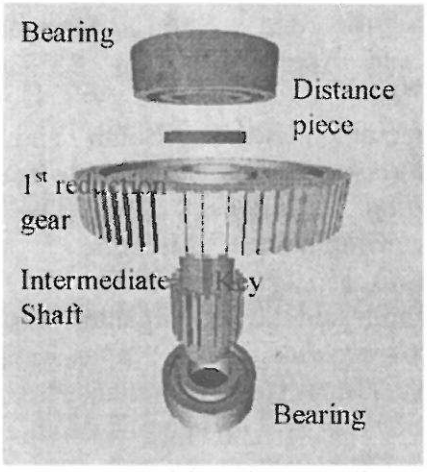


Figure 10: Initial intermediate shaft sub-assembly.

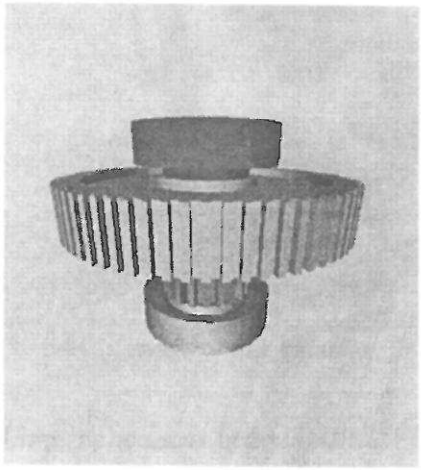


Figure 11: Final intermediate input shaft sub-assembly.

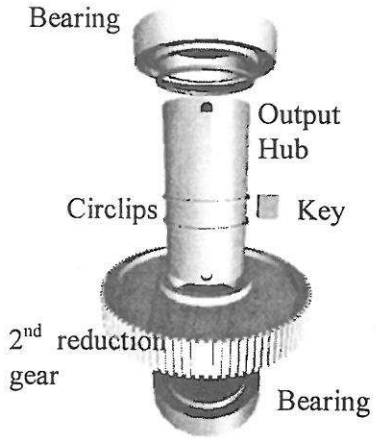


Figure 12: Initial output hub sub-assembly.

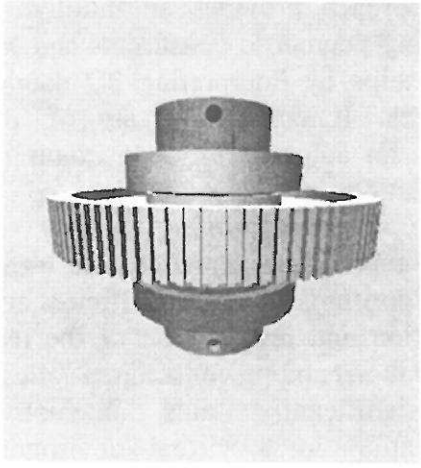


Figure 13: Final output hub sub-assembly.

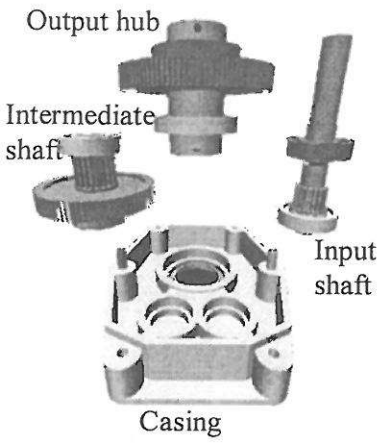


Figure 14: Initial speed reducer assembly.

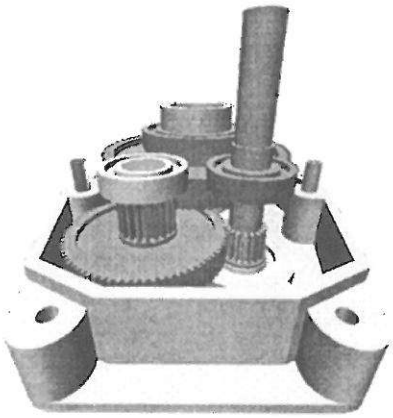


Figure 15: Final speed reducer assembly.

5. Conclusions and Future Work

This paper presents a new constraint-based 3D-manipulation approach that integrates direct 3D manipulation and 3D constraints. This approach provides an intuitive way of specifying geometric constraints and assembly relationships by interpreting 3D manipulation sequences. It employs a set of allowable motions for automatically constraining the 3D manipulation of solid models to achieve accurate 3D positioning simply by 3D manipulations. The approach also uses graph-based algorithms to perform efficient constraint satisfaction and propagation of the user's 3D manipulations among constrained solid models. This significantly aids interactive 3D manipulation in a virtual environment by relieving the user from the burden of ensuring constraint consistency during 3D manipulations. Industrial case studies such as the *puma robot* and the *shaft mounted speed reducer* are successfully demonstrated.

At present, investigation of a parallel algorithm for the constraint recognition process has been undertaken to achieve real-time response for complex solid models. In addition, research is being continued to develop incremental constraint satisfaction and propagation mechanisms for handling complex assembly mechanisms. Further research is underway at SIIT in collaboration with Asian Institute of technology (AIT) to exploit the work described in this paper and the grid generation methods [13] in building a virtual 5-axis milling machine simulation for the Maho600E 5-axis machine. The simulator may be built using the pre-defined constraints in which the allowable motion of each axis can be derived automatically. New graph-based algorithms will be developed to simulate the inverse kinematics of the 5-axis machine [14]. The output from the simulator may be used to estimate an error, handle the tool interference and identify the tool shape before actually testing with the real machine.

References

[1] R. Sodhi and J.U. Turner, "*Towards modeling of assemblies for product design*", CAD, V26, N2, pp. 85-97, February 1994.

[2] G.M. Bayliss and A. Bowyer and R.I Taylor and P.J Willis, *"Virtual Manufacturing"*, CSG'94: Set-theoretic Solid Modelling Techniques and Applications, pp. 353-365, April 1994.

[3] M. Figueiredo and K. Bohm and J. Teixeira, *"Precise Object Interactions using Solid Modelling Techniques"*, Modelling in Computer Graphics: Methods and Applications, pp. 156-176, June 1993.

[4] M. J. Papper and M. A. Gigante, *"Using Physical Constraints in a Virtual Environment"*, Virtual Reality Systems, Ed. by R. A. Earnshaw, M. A. Gigante and H. Jones, 1993.

[5] W. Sohrt and B. Bruderlin, *"Interaction with Constraints in 3D Modelling"*, International Journal of Computational

Geometry and Applications, V1, N4, pp. 405-425, 1991.
[6] M. Gleicher, *"Integrating Constraints and Direct Manipulation"*, Symposium on Interactive 3D Graphics, pp. 171-174, 1992.
[7] S. R. Alpert, "Graceful Interaction with Graphical Constraints", IEEE Computer Graphics and Applications, pp. 82-91, Mar 1993.
[8] E. A. Bier, *"Snap-Dragging in Three Dimensions"*, Symposium on Interactive 3D Graphics, pp. 193-204, 1990.
[9] Mingxian Fa, *"Interactive Constraint-based Solid Modelling"*, Phd Thesis, School of Computer Studies, University of Leeds, 1993.
[10] EDS Unigraphics Division, *"Unigraphics system software with translator V 10.2 Release"*, Feb 1994.
[11] M. Fa and T. Fernando and P. M. Dew, "Interactive Constraint-based Solid Modelling using Allowable Motion", Proc. of ACM/SIGGRAPH Symposium on Solid Modelling and Applications, pp. 243-252, May 1993.
[12] A. A. G. Requicha and J. R. Rossignac, "Solid Modelling and Beyond", IEEE Computer Graphics and Applications, V12, N5, pp. 31-45, Sep 1992.
[13] S. S. Makhanov, *"An Application of the Grid Generation Techniques to Optimize a Tool-Path of Industrial Milling Robots"*, J. of Computational Mathematics and Mathematical Physics, V39, N9, pp.1589-1600, 1999.
[14] M. Munlin and S.S Makhanov. "Constraint-Based Simulation of a 5-Axis Milling Machine". "Proc. Of International Conference on Production Research", August 2000.