A Geometric Reasoning Server with Applications to Geometric Constraint Satisfaction and Reconfigurable Feature Extraction

Oscar E. Ruiz S., Rodrigo A. Marin, Placid M. Ferreira

Department of Mechanical and Industrial Engineering

University of Illinois at Urbana-Champaign

1206 West Green St. Urbana, IL, 61801, USA

Abstract

Geometric Reasoning can be defined as the modeling and manipulation of geometric entities and their relations. A large number of applications in Computer Aided Design, Manufacturing and Process Planning intensively manipulate and reason about Geometric Objects. However intensive is the use of geometrical utilities in a CAD/CAM/CAPP environment, the construction of such systems frequently uses an ad hoc set of geometric routines written for the particular application at hand. If another application is to be developed, the geometric requirements of it are satisfied in similar manner, therefore leading to a large replication of effort, a difficulty in their integration, and to difficulty in the modeling of robust and yet flexible and extensible applications. In this investigation, the design, modeling and development of a Geometric Reasoning Kernel which serves the needs for CAM/CAM/CAPP environments is discussed.

Geometric Reasoning acts in two conceptually different levels: the Static level in which fully instanced geometric objects and relations among them are manipulated. Examples of this sort of services are *logical queries*, which test sets of objects for relations among them (parallelism, perpendicularity, inclusion, etc), and *construction queries*, which build non ambiguous entities to meet certain requirements, for example construction of convex hulls, projections, object intersections, etc. In Dynamic Level, objects are specified by geometric relations, called constraints, which involve other objects in the World, possibly not completely defined themselves. The result is a system of interdependencies which frequently results in ambiguously or inconsistently defined scenarios. The evaluation of inconsistencies, ambiguities, and the production of a feasible world is called the Geometric Constraint Satisfaction or Scene Feasibility (GCS/SF) problem. A Static Geometric Reasoning Server has been developed using an object oriented approach to model geometric objects. The server is built in a way that it can be used as a library to which client programs can link, even providing them with services that allow the user to interact through a graphic interface on which the objects can be directly manipulated and visualized. Two (client) applications of the static geometric reasoning server have been explored. The first is a Reconfigurable Feature Extraction system. The second is the Dynamic Reasoning part itself. The core of the Reconfigurable Feature Extraction system is a high level language that allows the user to define private, customized features in terms of topological and geometric objects and relations. No generic algorithm is used to recognize the features. Instead, the language also allows the user to dynamically specify and automatically produce the recognition procedure. This makes possible to have an open library of features and to exploit the specific knowledge that users have about them and the context in which they are to be found. The conceptual development and theoretical foundations for the Dynamic Reasoning module are discussed here, and as a result, an algorithmic solution for the GCS/SF problem is presented. This algorithm heavily relies on concepts of algebraic geometry which allow to evaluate the characteristics of the solution space (possible scenarios) for the GCS/SF problem.

1 Introduction

Geometric relations are of primary importance in Computer Aided Design, Manufacturing and Process Planning. The ability to create, modify, maintain and reason about such relationships has to be provided to any software application in these areas. These abilities have traditionally been imparted to the software on a case-by-case basis. The repercussions of this approach are : (i) a failure to address the common, fundamental problems underlying particular instances, (ii) restricted domains of solutions, and (iii) replication of code, producing larger and difficult-to-maintain systems. In Spatial Reasoning, two main areas can be identified; *Static* and *Dynamic* Reasoning. *Static* Reasoning problems are those concerned with fully and unambiguously defined entities. Typical problems include *boolean* queries testing a particular relation among entities and *construction* queries which create entities satisfying relations with other given entities in the world, in general producing well defined (although not unique) answers. In *Dynamic Reasoning*, a set of geometric entities is specified by geometric relations (also called constraints) in the context of a given world. As the set of relations varies, the position (and the very nature) of the entities in the space changes. The specification may be ambiguous, resulting in an infinite number of possible answers, or inconsistent, producing an empty solution space. The determination of a set of geometric entities that satisfy a series of geometric relations (constraints) in the context of a basic, fixed world constitutes the Geometric Constrain Satisfaction or Scene Feasibility (GCS/SF) problem.

In this work, a centralized kernel philosophy is pursued with the objective of serving all of these levels of spatial reasoning, including solutions for instances of the GCS/SF problem while avoiding the drawbacks mentioned above.

This article presents the philosophy and implementation of a Geometric Reasoning (GR) Server, followed by two aspects of its application; first, the design and implementation of a client system which allows reconfigurable feature definition and extraction by using the capabilities of the Static GR module, and second, the theoretical fundaments and algorithmic support for solutions of the GCS/SF problem, which are the basis for the Dynamic GR module. Examples of application of this theoretical framework are discussed, which show different approaches to formulate and solve the GCS/SF problem, and evaluation of their advantages is presented here.

The applications of these capabilities spread over diverse fields in design and manufacture such as mechanism design, feature extraction, fixturing, assembly planning, parametric design, tolerancing analysis, etc.

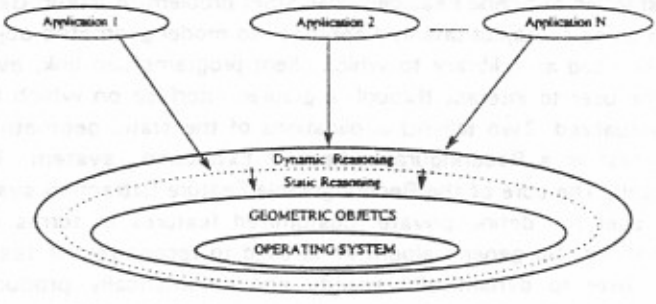

Figure 1. Centralized Geometric Server.

Figure 1 shows the underlying philosophy in the creation of a centralized Geometry Server. From the point of view of the user (programs or humans), it extends the capabilities of the computer, presenting geometric objects as primitive types in the computer collection of manageable objects, similar to real or integer numbers, or character strings. On top of this layer, the GR Server is anchored. The nature of the problems aimed at implies that

- Allow the user to use prior knowledge on the nature of the features and of the parts on which they will be recognized, to define the recognition strategy. This will put the correctness and the efficiency of the recognition process under direct user control, and these can be adjusted at his convenience.

We claim to have met these two requirements by having developed a high level language in which the user can conveniently define the features and the corresponding recognition strategies. The feature definitions in this language are compiled into a C++ program and linked with the Static GR Library and with the Application Programming Interface of the ACIS Solid Modeler. A brief description of the characteristics of the language is given in the sections below.

3.1.1 Feature Definition

Features are modeled as sets of attributed topological entities among which a set of topological and geometric relationships are defined. Topological entities are faces, edges and vertices. The topological relationships between the entities are defined by the connectivity existing between them. This can be assimilated to the frequently used Face Adjacency Graph representation. See Figure3.

The following geometric geometric attributes can be defined based on topological entities:

- *Faces geometric type:* Planar, cylindrical.
- *Edges geometric type:* Straight, circular, etc.
- *Edges convexity:* Edges can be convex or concave.

Geometric relationships between entities define the relative positioning and orientation. These can be perpendicularity, parallelism, coplanarity, colinearity, distance, angle, etc. The assessment of these relations is performed by using the the GR server discussed above.

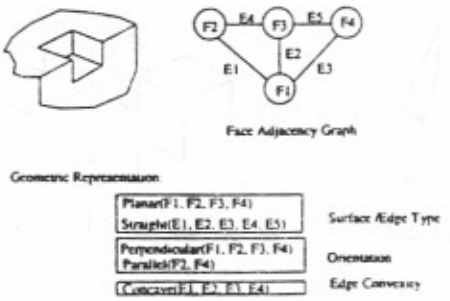

Figure 3. Definition of a Notch Feature

3.1.2 Feature Recognition Process Definition

The recognition of a feature is closely related to its definition, as described in the previous section. An instance of a solution to the recognition process is a one to one mapping between the topological entities given in the definition and the instances of topological entities found in the solid model of the part. The recognition process consists of finding the sets of instances of topological entities that satisfy the attributes and relationships given in the definition.

Two types of statements are possible in the recognition process specification:

- *Search Statements:* These statements try to instantiate one or more topological entities according to one or more criteria expressed in terms of attributes and/or relationships given in the definition and that must be satisfied by the entities.
- *Verification Statements:* These statements enforce the satisfaction of some attribute and/or relationship on one or more of the entities found in the partial solutions found so far. The instances not satisfying the criteria are removed.

3.1.3 Efficiency considerations

From the above discussion, it can be seen that if a certain set of entities in the solid model of the part are a valid instance of the feature to be recognized, it may happen that by permuting some or all of the entities in the set the feature definition will also be satisfied. This means that it is possible to have multiple solutions representing the same feature instance in the part. For example, if a slot is defined as a base face perpendicular and connected to two parallel side faces, a permutation of the two side faces in an instance of the slot would also be a valid solution. This can become a serious problem because of the risk of combinatorial explosion in the solution set.

Currently, the language is being enhanced to provide mechanisms to rule out the inclusion of permutations of the same feature in the solution set. The user will be able to specify what permutations are to be allowed or not at a particular stage of the recognition process. However, tests have shown that careful writing of the recognition procedure plays a key role in keeping the solution set as small as possible. Once a set of entities is instantiated using a search statement, it is a good practice to use one or more verify statements that will eliminate invalid solutions, before using additional search statements that increase the size of the solution set. The difference can sometimes be dramatic as demonstrated by the examples that follow.

3.1.3 Example

Two examples of a recognition procedure for a simple slot feature in a part are shown in this section (See Figure 4). These examples intend to show the use of the language in the recognition of a feature and to illustrate the efficiency considerations made in the previous section.

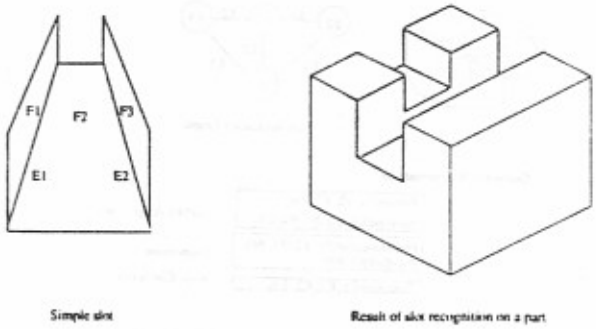

Figure 4. Simple slot feature

The first procedure is the following:

```
# First example of recognition program for a slot.

DEFINITION
    NAME SLOT ;

    ENTITIES:
            F1, F2, F3: FACE ;
            E1, E2, E3: EDGE ;
    EDGE TYPE:
            E1, E2: STRAIGHT ;
    FACE TYPE:
            F1, F2, F3: PLANAR ;
    CONNECTIVITY:
            F1:(F2, E1, CONCAVE) ;
            F2:(F1, E1, CONCAVE), (F3, E2, CONCAVE) ;
    ORIENTATION:
            F1, F3: PARALLEL ;
            F1, F2: PERPENDICULAR ;
            F2, F3: PERPENDICULAR ;
END DEFINITION

BEGIN
    FIND [E1, E2] BASED ON EDGE CONVEXITY ;
    FIND [F1] BASED ON CONNECTIVITY WITH [F2] ;
    FIND [F2] BASED ON CONNECTIVITY WITH [F3] ;
    VERIFY [E1] DERIVED CONNECTIVITY FROM [F1, F2] ;
    VERIFY [E2] DERIVED CONNECTIVITY FROM [F2, F3]
END
```

This first procedure is very inefficient, having to consider a set of 4128 candidates for the feature after the third recognition statement is executed. In total, the recognition procedure takes 140.23 seconds to produce the final result. The inefficiency is mainly due to the fact that all the search statements are executed before any verification statement, allowing the solution set to grow very large.

The second procedure is the following:

```
# Second example of recognition program for a slot.

DEFINITION
    NAME SLOT ;

    ENTITIES:
            F1, F2, F3: FACE ;
            E1, E2, E3: EDGE ;
    EDGE TYPE:
            E1, E2: STRAIGHT ;
    FACE TYPE:
            F1, F2, F3: PLANAR ;
    CONNECTIVITY:
            F1:(F2, E1, CONCAVE) ;
            F2:(F1, E1, CONCAVE), (F3, E2, CONCAVE) ;
    ORIENTATION:
```

```
                F2, F3: PERPENDICULAR ;
END DEFINITION

BEGIN
    FIND [F1] BASED ON CONNECTIVITY WITH [F2] ;
    FIND [E1] BASED EDGE CONVEXITY ;
    VERIFY [E1] DERIVED CONNECTIVITY FROM [F1, F2] ;
    FIND [F3] BASED ON CONNECTIVITY WITH [F2] ;
    FIND [E2] BASED ON EDGE CONVEXITY ;
    VERIFY [E2] DERIVED CONNECTIVITY FROM [F2, F3]
END
```

By contrast with the first example, this procedure performs verifications as early as possible between search statements. As a result, the solution set grows to a maximum of 288 elements and takes 3.5 seconds to execute.

3.1.4 Evaluation and Future Work:

Initial testing of this system has been completed. The system has been found to be robust and the flexibility provided by the language adequate. Currently the language is being enhanced to increase its expressive power and to improve its efficiency preventing the potential combinatorial explosion that can be caused by permutations of the entities defining a feature.

4 Theoretical Fundaments for Dynamic GR

Prior sections of this paper introduced the GR system in its *Static* part, and showed an application area in which the services of the system are used in a Reconfigurable Feature definition and Recognition client program. In this section, the theoretical foundations which allow us to attack the Dynamic Reasoning problem are laid out. Applications of this theory have been successfully realized in Assembly Planning and Kinematic Analysis of Mechanisms among other areas.

4.1 Problem Statement

The Dynamic Reasoning are essentially a version of the Geometric Constraint Satisfaction, or Scene Feasibility (GCS/SF) problem defined as follows: Let a World W be a closed, homogeneous subset of E^3, and a set of zero curvature geometric entities $S=\{e1,..en\}$ (points, straight lines, planes). A set of spatial relations between pairs of entities $R=\{R_{i,j,k}\}$ are specified, where $R_{i,j,k}$ is the k^{th} relation between entities i and j. The goal is to find a position for each entity e_i in the world W, $R_{i,W}$, consistent with all relations R specified on it. The world W is a basic, fixed scenario in which the entities S satisfying R will eventually be instantiated. One says that *S is feasible for W and R*, and denotes this fact by $S=feasible(W,R)$. This problem can be translated into the solution of a set of polynomial equations $F = \{f_1, f_2, ...f_n\}$; therefore *F is the polynomial form of the problem (W,R)*; and it is written as $F=poly\_form(W,R)$. If S is a solution for F, we write it as: $S=solution(F)$.

4.2 Modeling Methodology

The terms used are explained next; *entity* means *geometric entity*: point, line or plane. Each entity has an attached frame. Points are in the origin of their attached frame. Lines coincide with the X axis of their frame. Planes coincide with the Y-Z plane of their attached frame. The world W contains a set of topological (polyhedra and possibly non manifold objects) and geometrical (lines, planes, points) entities $S=\{e_1,e_2,..e_n\}$. For the discussion at

hand it is assumed that the entities are part of a body. F_{ij} is the known, fixed position of entity i with respect to the body frame j. R_k represent relations or constraints between entities. These relations, shown in Table 2 represent *contact* relations which can be stated as "place entity E_1 ON entity E_2" (E_1-*ON*-E_2), and they can be expressed in the vector form shown there.

Table 2: Constraint Relations and Polynomial Forms)

relation	*entity 1*	*entity 2*	*vector equation*
P-ON-P	p_1	p_2	$p_1 = p_2$
P-ON-LN	p_1	$LN=(p_2,v_2)$	$(p_1 - p_2) \times v_2 = 0$
P-ON-PLN	p_1	$PLN=(p_2,n_2)$	$(p_1 - p_2) \cdot n_2 = 0$
LN-ON-LN	$LN=(p_1,v_1)$	$LN=(p_2,v_2)$	$v_1 \times v_2 = 0$
			$(p_1-p_2) \times v_2 = 0$
LN-ON-PLN	$LN=(p_1,v_1)$	$PLN=(p_2,n_2)$	$(p_1-p_2) \cdot n_2 = 0$
			$v_1 \cdot n_2 = 0$
PLN-ON-PLN	$PLN=(p_1,n_1)$	$PLN=(p_2,n_2)$	$(p_1-p_2) \cdot n_2 = 0$
			$n_1 \cdot n_2 = \pm 1$

4.3 Grobner Basis

Each relation (constraint) R_i enforced on the entities of the World translates into a number of polynomial equations as discussed above. The fact that there is a Feasible World for that set of constraints is directly determined by the existence of simultaneous roots to the set F of polynomials originated from all the constraints imposed on the (entities of the) system. The analysis of the common roots for this set can be performed in the framework of Algebraic Geometry Techniques. In particular, an alternative set of polynomials called Grobner Basis, (*GB(F)*) can be analyzed. In the present work some properties of the Grobner Basis of the set F will be stated with no proof or further discussion. We are assuming that the set $F=\{ f_1, f_2, f_3, \ldots, f_n \}$ has variables $x_1, x_2, \ldots x_n$ for which we define an ordering $x_1 < x_2 < \ldots < x_n$. For a deep treatment of this topics see [3,2,1].

Proposition 1. $S=solution(F)$ iff $S=solution(GB(F))$.

Proposition 2. *1* in $GB(F) \Rightarrow F$ has no solution.

Proposition 3. These exists an algorithm *ZeroDimensional(GB(F))* which is able to decide whether F has a finite number of zeros. In such a case $S=feasible(W,R)$ has a finite number of configurations.

Proposition 4. There exists an algorithm *In_Radical(GB(F),f)* which allows to decide whether the zeros of F are contained in the zeros of f. In that case, f expresses a constraint already included in F.

Proposition 5. There exists an algorithm *GB(F)* which produces a *triangular* Grobner Basis, in the sense that *GB(F)* contains polynomials only in x_1, some others only in x_1, x_2, and so on, making the numerical solution a process similar to triangular elimination.

4.4 An Algorithmic Solution to the GCS/SF Problem

This theoretical background can be summarized in the following macro-algorithm, in which the invariant clause for the loop is the existence of a set of non-redundant, consistent and multi-dimensional set of (constraint generated) polynomials. In the event of the addition of new constraints to the scene, the algorithm converts them into polynomial(s), and tests their redundancy (by using Proposition 4), inconsistency (Proposition 2) and Zero Dimensionality (Proposition 3). If the new constraint is redundant no action is taken; in the other two cases

the invariant becomes false and the loop breaks. If the ideal has become Zero-dimensional a triangular Grobner Basis under some stated order is extracted (Property 5) and solved. Proposition 1 is the underlying basis of the algorithm, since it establishes that the *GB(F)* faithfully represents F, with the same roots and ideal set.

```
                    {Pre: W a fixed scenario }
                    F = {}
                    GB = { }
                    do  new relation R_i
                            {Inv:  F is consistent, non redundant,, Multi-dimensional }
                            R = R + { R_i }
                            f = poly_form( W, R_i )
Proposition 2               if  (1 in GB( F  +  {f} ) ) then
                                    stop ( system is inconsistent )
                            else
Proposition 4                       if   In_Radical(GB(F),f)  then
                                            skip ( f is redundant )
                                    else
                                            F = F  + { f }
Proposition 1                               GB = GrobnerBasis(F, prec )
Proposition 3                               if  ( ZeroDimension(GB)) then
                                                    break loop
                                            else
                                                    skip (next relation-constraint)
                                            fi
                                    fi
                            fi
                    od
Proposition 5       S = triangular_solution( GB )
                    {Post:R = {R_i} a set of relations; S = feasible(W,R) }
```

4.5 Example

In order to illustrate the methodology for modeling and solution of the GCS/SF problem a very simple example is presented here, which follows the methodology applied to larger and more complex systems ([4,5]). Consider a scene in which there are two straight lines $LN_1 = (P_1, v_1)$ and $LN_2 = (P_2, v_2)$ (See Figure 5) expressed parametrically, and assumed to be rigidly linked to each other by a displacement M. Another set of lines, with similar conditions are given by $LN_3 = (P_3, v_3)$ and $LN_4 = (P_4, v_4)$.

The proposed relations place LN_1- *ON* -LN_3 *and* LN_2- *ON* -LN_4 *(being* LN_3,LN_4 also rigidly joined). The goal of the problem is to find whether the relations can be satisfied, what displacement is to be performed on the rigid body holding LN_1 and LN_2 to achieve the goal, and the degrees of freedom that are afforded to the body holding LN_1 and LN_2 by the relationship. In this example, an abstraction of an electric outlet plug, the remaining variable is a translational degree of freedom in the direction of the line axes.

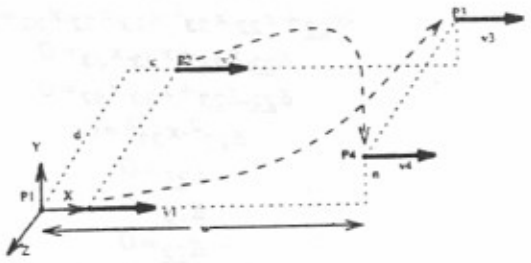

Figure 5: Simultaneous Line-to-Line Restriction between Pairs of Lines

The problem can be stated as follows:

1. Apply a (still unknown) rigid displacement D to LN_1 and LN_2. D is formed by a rotation Rot and a translation T.

$$Rot = \begin{matrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{matrix} \quad ; \qquad T = \begin{matrix} T_x \\ T_y \\ T_z \end{matrix}$$

The transformed entities are

$$P_1' = T + Rot.P_1;\ v_1' = Rot.v_1;\quad P_2' = T + Rot.P_2;\quad v_2' = Rot.v_2$$

2. The specified relations (*parallel, contained,* etc) impose the following conditions (expressed in vector terms for simplicity):

$$\begin{matrix} (P_1' - P_3) \times v_3 = 0; & P_1' \text{ in } LN_3 \\ v_1' \times v_3 = 0; & v_1' \,||\, v_3 \\ (P_2' - P_4) \times v_4 = 0; & P_2' \text{ in } LN_4 \\ v_2' \times v_4 = 0; & v_4' \,||\, v_4 \end{matrix}$$

$$det(Rot) = +1;$$

The condition $det(Rot) = +1$ imposes dexterous orthonormality to the matrix $Rot = [v_1:v_2:v_3]$. Orthonormality implies $|v_i| = 1,\ (i = 1..3);\ ::v_j.v_i = 0,\ (i \neq j)$. Dexterity implies $v_1 \times v_2 = v_3$.

The equations arrived at are polynomials, whose solution determine the matrix D, and therefore the position of the pair LN_1-LN_2. The analysis of this set of equations is performed within the framework provided by the algorithm above.

The (lexicographic) order used in this example, for the calculation of the Grobner Basis is: $x_{11} > x_{12} > x_{13} > x_{21} > x_{22} > x_{23} > x_{31} > x_{32} > x_{33} > T_x > T_y > T_z$.

When the first constraint is applied (LN_1-ON-LN_3), the conditions

$$\begin{matrix} (P_1' - P_3) \times v_3 = 0; & (P_1' \text{ in } LN_3); \\ v_1' \times v_3 = 0; & (v_1' \,||\, v_3) \end{matrix}$$

produce an solution:

$$d + T_z = 0 \quad n - T_y = 0 \quad x_{31} = 0 \quad x_{21} = 0$$

The Grobner Basis corresponding to this condition is shown below. The $\underline{f}$ notation is used to identify the variables which are the leading terms in polynomial f:

$$\underline{T_z} + d = 0$$
$$\underline{T_y} - n = 0$$
$$\underline{x_{32}}^2 + x_{33}^2 - 1 = 0$$
$$\underline{x_{31}} = 0$$

$$\underline{x_{23}}^2 + x_{33}^2 - 1 = 0$$
$$-\underline{x_{22}} + x_{22} \cdot x_{33}^2 - x_{23} \cdot x_{32} \cdot x_{33} = 0$$
$$\underline{x_{22}} \cdot \underline{x_{32}} + x_{23} \cdot x_{33} = 0$$
$$\underline{x_{22}} \cdot \underline{x_{23}} + x_{33} \cdot x_{32} = 0$$
$$\underline{x_{22}}^2 - x_{33}^2 = 0$$
$$\underline{x_{21}} = 0$$
$$\underline{x_{13}} = 0$$
$$\underline{x_{12}} = 0$$
$$\underline{x_{11}} + x_{23} \cdot x_{32} - x_{22} \cdot x_{33} = 0$$

The parameters of the World configuration (c,d,w,n) appear as constants in the basis. First, the fact that it does not contain 1, suggests that we cannot conclude that the constraint is inconsistent with the preexisting scene. Yet, we might have a solution with complex variables which is not physically realizable. Second, the *ZeroDimensional()* algorithm of proposition 3 (see [3]) establishes that T_x and x_{33} stay as free variables, therefore producing a two dimensional space of solutions. In T_x and x_{33} we can recognize a translational and rotational degrees of freedom respectively,

Suppose, the second constraint (LN_2-*ON*-LN_4) resulting in the equations

$(P_2' - P_4) \times v_4 = 0$; $(P_2'$ *in* $LN_4)$; $v_2' \times v_4 = 0$; $(v_2' \,||\, v_4)$

is added to the scene.

The Grobner Basis for the accumulated constraints, once again, shows neither inconsistency nor zero-dimensionality, for the same reason as before.

$$\underline{T_z} + d = 0$$
$$\underline{T_y} - n = 0$$
$$\underline{x_{33}} + 1 = 0$$
$$\underline{x_{32}} = 0$$
$$\underline{x_{31}} = 0$$
$$\underline{x_{23}} = 0$$
$$\underline{x_{22}}^2 - 1 = 0$$
$$\underline{x_{21}} = 0$$
$$\underline{x_{13}} = 0$$
$$\underline{x_{12}} = 0$$
$$\underline{x_{11}} + x_{22} = 0$$

In this case, however, T_x variable is effectively the only degree of freedom left. Two assembly modes are possible, by setting $x_{22} = \pm 1$.

If an additional constraint is set, for example P_1-*ON*-P_3, the Grobner Basis becomes Zero Dimensional, reflecting the fact that all the degrees of freedom are now fixed, and there are a finite number of configurations (*D* transformations) to satisfy the conditions:

$$\underline{T_z} + d = 0$$
$$\underline{T_y} - n = 0$$
$$\underline{T_x}^2 - c^2 + w^2 - 2.w.T_x = 0$$
$$\underline{x_{33}} + 1 = 0$$
$$\underline{x_{32}} = 0$$
$$\underline{x_{31}} = 0$$
$$\underline{x_{23}} = 0$$
$$c.\underline{x_{22}} + w - T_x = 0$$
$$\underline{x_{21}} = 0$$

$$\underline{x}_{13} = 0$$
$$\underline{x}_{12} = 0$$
$$\underline{x}_{11} \cdot c\text{-}w + T_x = 0$$

If yet one more condition is set, unless it is redundant, the system becomes inconsistent; for example, if one requires P_2-*ON*-P_4, the Grobner Basis shows the inconsistency:

$$GB = \{ 1 \}$$

The modeling of the GCS/SF problem and its solution using the techniques discussed above has been implemented and have shown that although the computational expenses incurred by the Grobner Basis calculation (*GB = GrobnerBasis(F, prec)*) are high, there are special sets of variables, derived from the formulation of the GCS/SF problem in terms of subgroups of the Euclidean Group *SE(3)* which induce better performance in the algorithm. These results are omitted in this abstract owing to the limited space available.

5 Conclusions

A Geometric Reasoning (GR) server has been described, which presents several advantages in terms of simplification of software production, reusability of highly specialized knowledge, compatibility for data structures and algorithms, accessibility from a client and expandability. The server has been implemented and its *Static* Reasoning Capabilities are used in a Reconfigurable Feature Extraction and Definition system. The theoretical foundations for the *Dynamic* GR system have been discussed, in the context of the GCS/SF problem, and an example has been provided. It is felt that *Dynamic* reasoning is the underlying framework for many problems in CAD/CAM, Mechanism Analysis, Robotics, Tolerancing, and other design and manufacturing applications.

References

[1] B. Buchberger. Applications of Grobner Basis in Non-Linear Computational Geometry. In D. Kapur and J. Mundy, editors, *Geometric Reasoning*, pages 413-446. MIT Press, 1989.

[2] Christoph M. Hoffmann. *Geometric and Solid Modeling*. Morgan-Kaufmann Publishers Co., 1989.

[3] D. Kapur and Y Lakshman. Elimination Methods: An Introduction. In B. Donald, D. Kapur and J. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence*, pages 45-88. Academic Press, 1992.

[4] O. Ruiz and P. Ferreira. Spatial Reasoning for Computer Aided Design, Manufacturing and Process Planning. Technical Report UILU-Eng-94-4004, University of Illinois at Urbana-Champaign, 1994.

[5] O. Ruiz and P. Ferreira. Grobner Bases and Group Theory in Geometric Constraint Satisfaction. Proc. International Symposium on Symbolic and Algebraic Computation, Oxford, England , 1994.