SPATIAL REASONING
FOR
COMPUTER AIDED DESIGN, MANUFACTURING AND PROCESS PLANNING

BY

OSCAR EDUARDO RUIZ

Mec. Ing., Universidad de los Andes, 1983
D. Ing., Universidad de los Andes, 1987
M.S. University of Illinois, 1991

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1994

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

DECEMBER 6, 1994

WE HEREBY RECOMMEND THAT THE THESIS BY

OSCAR EDUARDO RUIZ

ENTITLED SPATIAL REASONING FOR COMPUTER AIDED DESIGN, MANUFACTURING AND PROCESS PLANNING

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

P Ferreira.

Director of Thesis Research

Head of Department

Committee on Final Examination†

P Ferreira.

Chairperson

N. Kapoor

E. Graham Eppl

Udatha G. Pabla

† Required for doctor's degree but not for master's.

O-517

ACKNOWLEDGMENTS

I want to express my sincere appreciation and gratitude to my Ph.D. thesis advisor, Professor Placid M. Ferreira for his guidance, encouragement and support in the course of this work. In the moments of the greatest difficulties he always had useful suggestions. I thank Professor Shiv Kapoor for the help and understanding I received from him during my Masters and Ph.D. work. For this, I am indebted to him. Professor Udatta Palekar gave me advice that was both theoretically elegant and at the same time very applicable, whenever my research touched his areas of interest. Professor Graham Evans from the Mathematics Department of the University of Illinois provided extremely valuable material and technical insight in the area of Algebraic Geometry. I feel that I owe special recognition to Professor Richard DeVor, my co-advisor in the Masters program, for the encouragement, fairness and respect that he always had for me.

The example of the hard work and responsibility set by my parents Carlos and Teresa gave me strength and endurance in the hardest moments. The love for knowledge of my brothers and sisters helped me to understand the importance of my education.

I owe a great debt to my friends Carlos, Rosario, Dena, Carlos Varela, Julie, Luciana, Rodrigo, Ricardo, Eri and Marisol because they have taught me courage and friendship and have provided extremely joyful times for me. Unhappy individuals cannot produce research.

Dr. Charles Wu, Dr. Sin Tang and Perry MacNeil provided a nice environment for my work at Ford Motor Company, where I learned a lot. My thanks to them.

My friends in the Mechanical Engineering Department have been a source of joy and technical help: Robert, Vivek, Paul, Giorgio, Kimberly, Kin, Chris, Bashar, Bhaskar, Suresh, Gu, Pei, etc. I have had a very happy time with them during these years.

The members of the staff of the Mechanical Engineering Department, who always have helped me, deserve this recognition: Amanda Horner, Jeff Oberg, Bill Marble, Marian

Brinkerhoff, Amy Reinhart and Haydenia Jenkins. I also want to acknowledge the late Judi Dimmett's kindness to me.

It is my pleasure to thank faculty members from other institutions for their personal interest in my research: Carme Torras and Federico Thomas from Universidad Polytecnica de Catalunya, Barcelona, Spain; Uwe Jasnoch from Fraunhofer Institute for Computer Graphics, Darmstadt, Germany; Jacques Herve from Ecole Centrale, Paris, France; Jorge Angeles from McGill University, Montreal, Canada; Anath Fischer from Technion Israel Institute of Technology, Haifa, Israel; Robin Popplestone from University of Massachusetts at Amherst, and many others.

I am also grateful for the education I received from Universidad de los Andes, my alma mater in Colombia.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction and Literature Review

Geometric relations are of primary importance in Computer Aided Design, Manufacturing and Process Planning. Any software intended to serve these areas has to provide the ability to create, modify, maintain and reason about geometric relationships. These abilities have traditionally been imparted to the software on a case-by-case basis. Thus the general structure of the problem has been lost. The repercussions of this loss are: (i) a failure to address the common, fundamental problems underlying particular instances; (ii) a restriction in the application domains served; and (iii) a tendency towards software replication. These factors produce larger, difficult-to-maintain and economically unattractive systems. In this work, efforts towards improving this situation are undertaken, by producing a centralized kernel or server for spatial geometry tasks. This kernel can be accessed by several applications, and extended as needed. To be useful it has to be able to solve problems efficiently over a broad domain.

In a simple scenario, a spatial reasoning system should deal with geometric entities fully and consistently defined. Typical services in this case correspond to queries about orientation and position of entities relative to each other, inclusion of one geometric entity into another, unambiguous constructions based on the entities (e.g convex hulls), etc.

More complicated scenarios result from the creation of entities which hold prescribed relations with their surroundings. In some cases, the entities are fully specified by these relations. For example, calculating the convex hull of a tesselation of points produces a well defined object, a convex polyhedron. In other cases the desired entities could be ambiguously specified. For example an ambiguous specification might require the creation of a line perpendicular to another line, resulting in an infinite number of possible

answers. In yet other cases the relation specified between objects might be internally inconsistent making their creation/modification impossible. For example, an inconsistent specification might request a line being simultaneously perpendicular to two non-parallel planes. Producing a configuration of geometric entities which satisfies a set of spatial relations is called *Scene Feasibility* or *Geometric Constraint Satisfaction* problem.

This work pursues a centralized kernel philosophy with the objective of serving all of these levels of spatial reasoning, including solutions for instances of the Scene Feasibility problem.

1.1 Motivation and Objectives. Problem Definition

The following are examples of diverse fields in design and manufacture where the capabilities mentioned above find application:

Feature extraction is a process by which a subset of the geometric (solid) model of a body is identified as representing a feature. The process of feature extraction implies a search across the data structure of the solid model for a configuration of entities [18, 24] which satisfies relations inherent to the feature. The strategy used for the extraction (syntactic pattern recognition, graph matching, etc.) is essentially *connectivity* oriented [16]. However, the determination of the exact nature of the feature found involves the *geometry* of the model. For example, slots and protrusions are identical from the point of view of connectivity; they can only be distinguished by their geometry. To extract a given feature, the connectivity search must be complemented by a geometrical query system. Typical geometrical queries test for relations between entities such as perpendicularity, parallelism, distance, convexity, etc.

Fixturing concerns the holding of a workpiece during a manufacturing process. The assessment of the degrees of freedom of the workpiece implies the evaluation of orientation of supporting surfaces, positioning of center of gravity, projection of center of gravity on the supporting surfaces, 2D and 3D convex hull calculations, etc. Numerical calculations may be able to answer some queries (for example positioning of center of gravity). How-

ever, manipulation of *geometric concepts* is required to support the concept of degrees of freedom of the body.

Assembly planning is an area which intensively uses spatial reasoning capabilities. Information on perpendicularity, parallelism, distances between entities, etc is needed to validate/find a particular assembly scene [36, 34, 35, 37]. Again, these capabilities address only part of the problem. The problem of feasibility of an assembly implies a study of the possible relative motion between the bodies involved accompanied by an study of the dimensions and positions of the participant entities.

Parametric design uses the specification of relations among geometrical entities to facilitate the process of re-design in the event of changing specifications. Therefore any modifications to dimensions or positions in the design have to be compatible with the governing relations. Conversely, modification of the required relations has to be validated against the dimensions and positions of the existing objects. Finally, relations *and* objects may be required to change. Therefore, the software developer faces the challenge of keeping configurations which are consistent in the relational and geometrical aspects.

Tolerancing analysis requires the reasoning about perpendicularity, parallelism, consistency of dimensions, angularity, etc. These are central tasks of a Spatial Reasoning Module [32]. Additionally, issues such as inconsistent dimensioning and tolerancing are intrinsically Scene Feasibility problems.

The examples discussed demonstrate the need for having a centralized kernel for reasoning about spatial relationships. Such a kernel must: (i) contain tools allowing for queries and constructions for fully and consistently instantiated entities (*static reasoning*); and (ii) have the ability to create and reason about ambiguously defined scenes which are the result of spatial relations among their entities (*dynamic reasoning*). Since the design and planning process are sequentially specified, a tool is needed in order to detect inconsistencies in the specified relations, or else, to produce entities satisfying them.

As part of this work, a library will be developed to address the most common geometric queries and constructions present in *static* reasoning. This library also lends itself as a support for queries needed in the solution of the Scene Feasibility problem (*dynamic*

reasoning). Both modules are supported by a Geometric Modeling system, which is responsible for the data structures which represent the models of the world. Since the final goal is to characterize solutions for the instances of the Constraint Satisfaction problem, a theoretical work is presented which combines concepts from algebraic geometry, group theory and graph theory in order to make it computationally feasible. This theoretical work: (i) presents several techniques to express the problem in a tractable mathematical way; (ii) provides a solid theoretical background for the physical interpretation of mathematical results; and (iii) discusses the necessary abstractions and manipulations needed to improve performance of the computations. This work includes an application of the static library to tasks of Feature Extraction. It also applies the proposed techniques to analysis of constraint satisfaction to several domains in Design and Manufacture.

1.1.1 Research Objectives. The GCS/SF Problem

The objective of this work is to develop techniques to solve instances of the *Geometric Constraint Satisfaction or Scene Feasibility (GCS/SF)* problem. It can be stated as follows: Let a World W be, a closed, homogeneous subset of E^3, and a set of geometric entities $S = \{e1, ..en\}$ which are closed, connected subsets of W. A set of spatial relations among 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 solution to such a problem is constituted by either a diagnostic of *inconsistency* in the formulated relations, or an instance of a set of entities e_i in the world W consistent with all relations R specified on entity i.

As discussed in depth later, the GCS/SF problem requires for its solution the support of Dynamic and Static Reasoning systems. On the other hand, research conducted in specific application areas of reasoning [19, 36, 35] has shown that Static reasoning is also a necessary tool in solving the Dynamic reasoning problem. Following sections discuss the practical and theoretical aspects of the solutions to these problems. In the context of the GCS/SF problem the term *topology* will be used to refer to the spatial relations that the entities have to keep. The term *geometry* is related to the dimensions and positions of entities in the World.

Table 1.1 Common Geometric Algorithms

Name	*Specification*	*Size Measure*	*Complexity*
Planar Convex Hull	convex hull of set of planar points	N=number points	$O(N.log(N))$
3D Convex Hull	convex hull of set of points	N = number points	$O(N.log(N))$
Planar Point Inclusion	Point included in a polygon	N = number sides	$O(N)$
Convex Polygons Intersection	P,Q Polygons	Np, Nq number edges	$O(Np + Nq)$
Star Polygons Intersection	P,Q Polygons	Np, Nq number edges	$O((Np + Nq)^2)$
Kernel of a Polygon	P polygon	Np number edges	$O(Np)$

1.2 Components of a Geometric Reasoning Server

1.2.1 Static Reasoning Module

Static Reasoning problems are those concerning fully and unambiguously defined entities. Typical problems include *boolean* queries testing a particular relation among entities and *construction* queries which create new entities satisfying relations with other given entities in the world. Examples are classification problems, in which a point is tested for inclusion inside a polygon, construction of convex hulls, projections of rays onto objects, etc. These problems have well defined (although not unique) answers. For example, the perpendicular line from a point onto a curved surface is a set of well defined lines.

Static reasoning therefore spans a well defined (and complex) set of services, which are needed in common practice in environments of CAD / CAM. Table 1.1 presents a sample of common problems, along with their complexity [14]. A short browsing in $2D$ problems follows:

- **Given convex polygons P and Q, find the result of the intersection between them**. This type of algorithms exploits the fact that the intersection must be convex, and that its upper (lower) boundary is a merging of sections of the individual upper (lower) boundaries.

- A Star Polygon has an area whose points can "see" all the boundary, called *kernel*. The kernel is a convex polygon. The problem of **determining the kernel of a given star polygon** is equivalent to the intersection of N halfspaces in $2D$.

- **Find the set of polygons which form the intersection of two star polygons**. In this case, since the boundary of the two polygons P_q and P_p is described in consistent order, it is easy to determine the crossovers of boundaries and to form the collection of disjoint (convex) polygons which form the intersection.

From the examples above, it can be seen that although many problems have been deeply studied, static problems still present a rich variety and implementation challenges. These challenges are related to the optimization of algorithms, and the re-usability of parts of them in other problems.

1.2.2 Dynamic Reasoning Module

Dynamic Reasoning addresses the Scene Feasibility / Constraint Satisfaction (GCS/SF) problem. This problem can be stated in mathematical terms as a set of equalities and inequalities. Solving the GCS/SF problem is equivalent to the determination of the sets of values for the variables which solve the equations. This work will be limited to spatial constraints that can be expressed as equalities; for example *positioning* constraints. Other types of constraining conditions, for example enforcement of *non-invasive* positioning, are expressed in the form of inequalities. They are not within the scope of this investigation.

Solving the GCS/SF problem requires the answer to the following points:

(1) Calculate the configuration(s) which satisfy the given constraints.

(2) Are the given entities over or under-constrained?

(3) If it is under-constrained, how many degrees of freedom are still available?

(4) What is the relation between variables used in the mathematical form of the GCS/SF problem and physical degrees of freedom of the entities involved?

Question 1 has been partially answered with the help of numerical techniques [28, 27, 1, 8]. However, they produce a particular answer, even in cases in which the degrees of freedom form a set of either infinite or finite number of solutions. Furthemore, to determine a particular answer the methods are limited; failure of the numerical method to produce an answer might not mean an empty solution space (over-specification in the problem), but instead a convergence problem in the implementation of the numerical procedure. Therefore numerical techniques, although needed for determining particular configurations, do not answer the questions 2, 3, and 4.

Questions 2-3 have not been satisfactorily answered in a systematic manner to the present time, basically because the dimension of the solution space for the GCS/SF problem is a function of topological *and* geometrical conditions. In other words, manipulation of the topological part of the GCS/SF problem does not suffice in determining the topology (degrees of freedom) of the solution space. This work explores several techniques used in order to answer questions 2 and 3. They include Grobner Bases [21, 7] and Characteristic Sets [10, 9]. The Characteristic Set method is mainly used for automatic theorem proving; given a set of hypotheses in the form of polynomials $H = \{h_1, h_2, ..h_n\}$ and a conclusion polynomial c, the method establishes whether or not c follows from H, and extracts degenerate conditions under which c does not necessarily follow from H. Application of this method to problems of spatial reasoning requires the statement of a conclusion [20]; which, in the case of GCS/SF problems, would require a hypothesis about the solution scenario. This is usually not available. Also, if the response of the algorithm is that conclusion c does not necessarily follow from H, no additional information is gained. These reasons have led to an exploration of methods other than Characteristic Sets to deal with the questions asked.

Grobner Bases are a consistent and computable way for reasoning about the existence of solutions to a set of polynomials. They also provide a framework in answering questions about the dimensionality of the solution space. This investigation will show that this theory helps to address questions 2 and 3 above. However, since this background has been forged in the domains of pure algebraic geometry, its specific application to the GCS/SF

problem has not been intended. In response to this situation, this work formalizes the analysis of the GCS/SF problem by using Algebraic Geometry theory. In this way, properties of Grobner Bases will help to resolve issues about the number of feasible geometric scenarios, inconsistency and redundancy of constraints, etc.

In the literature reviewed, question 4 has been approached in special cases from the areas of kinematics, mechanisms and *group theory* [19, 2, 3]. A joint in a rigid bar mechanism is, by definition, a constraint. Therefore, historically, the study of mechanism analysis precedes constraint satisfaction problems. At the same time, a mathematical abstraction of constraints is provided by group theory. This multiplicity of disciplines studying the same area is manifested in the fact that the terms *(trivial) constraint, joint* and *group* are used interchangeably in the discussion. The techniques described in [19, 2, 3] are limited to reasoning about *topology* of constraint networks, and to the special cases in which the constraints are of the *trivial* type (discussed later in this document). In this investigation, the theoretical background built for questions 1-3 will allow the integration of topological and geometrical analysis, without the limitation of *triviality* of constraints. In addition, problem formulations derived from the use of group theory will relate the variables used with the degrees of freedom of the participant entities.

The following sections present a theoretical background and review of related material. This review allows for a more specific and formal discussion of the research objectives.

1.3 Theoretical Background

Three theoretical aspects need to be addressed in this investigation: (i) Since the GCS/SF problem can be expressed in terms of sets of polynomials, its solution corresponds to the existence of common roots for the polynomials in the set. In this respect, Grobner Basis provides a background for analysis of the solution of the set of polynomials. (ii) Given that the $SE(3)$ group allows the efficient expression of the geometrical nature of the GCS/SF problem, an introduction to the relevance of group theory is presented. (iii) Since the Spatial Constraint graph is a pictorical representation of the GCS/SF prob-

Table 1.2 Elementary Relations and Polynomial Forms

Relation	*Arguments*	*Vector Form*	*Polynomial Form*
perpendicular	v,w vectors	$v.w = 0$	$v_1.w_1 + v_2.w_2 + v_3.w_3 = 0$
parallel	v,w vectors	$v \times w = 0$	$v_2.w_3 - v_3.w_2 = 0$ $v_3.w_1 - v_1.w_3 = 0$ $v_1.w_2 - v_2.w_1 = 0$
magnitude	v vector	$v.v = d^2$	$v_1.v_1 + v_2.v_2 + v_3.v_3 = d^2$

lem, it is felt that the decomposition of the SC graph has close relation with embedded subproblems of the GCS/SF problem. To address such issue, graph partition, selection of subproblems and integration of partial solutions into the general solution need to be explored.

1.3.1 Algebraic Geometry. Grobner Bases

Table 1.2 shows how relations commonly used in Spatial Reasoning translate into vector and polynomial equations. These equations use the variables corresponding to position and orientation of the entities involved. For example, the specification of two planes being parallel implies their normal vectors being parallel, forcing relations such as those shown in Table 1.2. Therefore, solution to a set of geometric constraints involves the analysis of the set of common roots for a polynomial set. This type of variables is called here, for reasons discussed later, *non-canonical*. Canonical variables will be introduced in following sections, and will carry considerable weight in this investigation.

There are several (symbolic) techniques [22] used to study the roots of sets of polynomials. Among them, Grobner Bases provide several characteristics which make it suitable for solving this kind of problems. In [21, 7], a discussion about the more fundamental facts in Grobner Bases theory and applications is available.

Given a set of polynomials F, which expresses the GCS/SF problem, a Grobner Basis for F, $GB(F)$, is a mathematically equivalent set, with convenient properties for root solving [21]. By examining $GB(F)$ one can draw conclusions about the nature of the roots in the original set F. In particular, Grobner Bases allow the answering of ques-

tions about existence of solutions (real or complex), the dimension of the solution space (empty, Zero-Dimensional or Multi-Dimensional) and the dependence (or redundance) of a new polynomial on (with respect to) polynomials existing in the set. It also presents advantages in determining a particular numerical solution.

The algorithm proposed by Buchberger [6, 7] for the construction of the Grobner Bases of a set of polynomials F does not take any advantage of particular characteristics of the application domains which produce the set F. Therefore, it is necessary to use formalizations that set up an efficient characterization of the GCS/SF problem. Such characterization would be then submitted for solution to Grobner Bases computation algorithms. The formalization of the GCS/SF problem in terms of Group Theory fulfills such a goal.

1.3.2 Euclidean Groups

Groups are sets of mathematical elements, furnished with a binary associative operation, displaying a neutral element and an inverse for every element in the set. In this investigation, the group of interest is $SE(3)$, the semi-direct product $R^3 \circ SO(3,R)$, where R^3 is the translational part, $SO(3,R)$ is the special orthogonal group, representing all right handed orthonormal 3-D frames and $\circ$ is the group multiplication operation. If G_1, G_2 and G_3 represent displacements in $SE(3)$ the (non-commutative) group properties are satisfied: (i) Two displacements G_1, G_2 applied in sequence produce a new displacement $G_3 = G_1 \circ G_2$. (ii) I is the null displacement in $SE(3)$; $G \circ I = I \circ G = G$. (iii) For each $G \in SE(3)$ there is an inverse displacement G^{-1} which undoes the displacement effected by G; i.e. $G \circ G^{-1} = G^{-1} \circ G = I$. (iv) The effect of displacements is accumulative. If G's are applied in the order G_1, G_2 and G_3, the following sequences are identical: $(G_1 \circ G_2) \circ G_3 = G_1 \circ (G_2 \circ G_3)$.

In [19], Herve was able to relate the structure of the group $SE(3)$ to the displacements allowed by kinematic joints. In this way formal structures were introduced (conjugation classes) which allow the naming of certain displacements in $SE(3)$ as "linear translations", "rotations", "planar slidings", etc. For example, a *rotational joint*, can

be described as the *rotational* class of displacements, written as $Ru(\theta)$ to stress the fact that it has *one*, *rotational* degree of freedom, θ, about the axis u. The prescribed kinematic relations are herein called *constraints*, since they restrict the degrees of freedom of the entities expressed by the conjugation class. The variables expressing the degrees of freedom of the joint (for example θ) in this formulation will be called *canonical*. The direct relation between constraints in the context of the GCS/SF problem and canonical forms of constraints as classes of the group $SE(3)$ can be exploited for the solution of the GCS/SF problem.

Herve, and later Thomas & Torras [36, 34, 35] used the fact that, as sets, groups can be composed and intersected. The composition reflects the sequential application of constraints, while the intersection reflects the simultaneous application of two constraints. From a kinematic point of view, several composed constraints can be thought of as a serial mechanism. Constraints intersected can be taken as several links (in parallel) sharing an end effector.

Sequences of constraints are expressed as:

$$R_1 \circ R_2 \circ R_3 .... \circ R_n \tag{1.1}$$

When the sequence has length 1, it is called *Trivial* constraint. In [19] Herve provides tables in which the results of composition and intersection of trivial constraints are tabulated. Non-trivial constraints, however, are present in many concepts in scene specifications; for example in the concepts of parallelism, distance and angles between entities, etc.

As mentioned before, the GCS/SF problem presents a strong inter-dependency between geometry and topology of constraints. The existing methods using the $SE(3)$ group [36, 34, 35] for constraint manipulations only address the topology of constraints. They also present limitation in that they only deal with trivial constraints. In this investigation, however, group theory is exploited to shift the emphasis in the modeling of the GCS/SF problem from the *positions* to the *degrees of freedom* of the entities involved.

The convenience of this alternative formulation will be assessed in this work. Additionally, the joint application of canonical variables in conjunction with the techniques of algebraic geometry will be investigated.

1.3.3 Graph Theory

The Graph of Spatial Constraints (see Figure 1.1) conveys the topological and geometrical information of the GCS/SF problem. The representation of the GCS/SF problem by a Spatial Constraint Graph allows a series of unexploited advantages: (i) a very clear formulation of the problem; (ii) a systematic way, suitable for computer applications, of generating the equations governing the degrees of freedom of the entities involved and; most importantly, (iii) the identification of subproblems which help in the solution of the GCS/SF problem, by allowing the application of preprocessing techniques. In order to have a clear convention for discussion ahead, the Graph of Spatial Constraints is introduced [35], using the following conventions:

(1) B_i: Represent the Bodies ($i = 1...n$).

(2) F_{ij} (frame of) the feature i in body j.

(3) C_i, $i = 1..m$: Constraint Relations

Since entities are represented by frames, in the following discussion the terms *entity* and *frame* are used as equivalent. In the SC graph, the nodes are *entity* frames (B_j and F_{ij}), while the arc between two nodes represents the *displacement* that relate the corresponding entity frames. Two entities may be joined by more than one arc, to admit more than one constraining relation between them. There are two types of nodes; nodes B_j represent (a position frame of) an entity in the World Coordinate System. Feature nodes F_{ij} represent the feature i in body (frame) B_j. The arcs in the graph represent displacements which convert coordinate frames.

Conceptually, there are two types of arcs: *positioning* and *constraint* arcs. Positioning arcs represent *known* relative positions of entities or features. They always join an entity

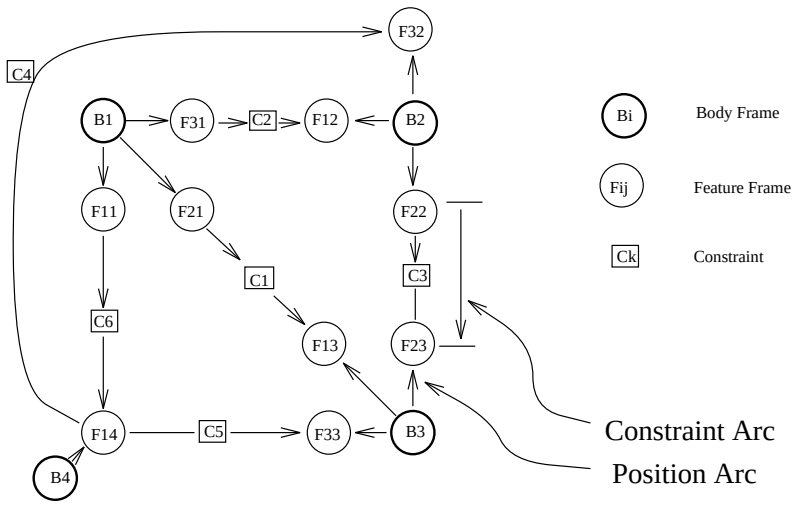

Figure 1.1 Spatial Constraint Graph

B_i and one of its features, F_{ji}. Abusing the notation this arc is named F_{ji} , as the feature itself. Constraint arcs always communicate two feature nodes, representing the degrees of freedom that a particular constraint allows. The constraint arcs are represented by $C_i(x_j, \theta_m, ...)$. In some cases the degrees of freedom $(x_j, \theta_m ...)$ are omitted for the sake of clarity. *Positioning* arcs express the geometry of the GCS/SF problem while *constraint* arcs represent its topological structure. Given a number of entities in the world and (trivial and non-trivial) constraints among them, the objective is to transform the Spatial Constraint Graph in such a way that constraints subgraphs are replaced by equivalent, simpler ones. The existing approaches [19, 36] present the following limitations in reaching such a goal: (i) the reduction process does not always reduce the constraint graph; (ii) only trivial constraints are handled; and (iii) being a topology oriented process, the geometric (or dimensional) part has to be dealt with separately.

Although the idea of preprocessing the loops of the SC graph appears in Thomas & Torras work [36, 35], no further elaboration on it has been presented. Another early application of graph formalism in mechanism analysis was presented by Dobrjanskyj & Freudenstein in [13]. Again, the analysis of local parts of the mechanism was not attempted, and therefore the decomposition of the kinematic graph was not pursued. This

investigation will address the influence of the SC graph decomposition on the analysis of local constraints, and on the solution of the full GCS/SF problem. In assembly planning, for example, such analysis is central to the mapping of local subgraphs of the SC graph into sub-assemblies. Treatment of sub-assemblies as clusters of bodies greatly simplifies the GCS/SF problem.

The identification of cycles of a graph has been studied in connection with several engineering applications. For example, in the analysis of electric networks the cycles in the graph of a circuit allow for the formulation of Kirchoff laws and similar techniques [33]. Two questions in such analysis present particular importance for our purposes: (i) how many cycles exist in a graph which convey independent information about the connectivity (topological structure) of the graph; and (ii) how to obtain a set of such non-redundant cycles which *completely* expresses the connectivity of the graph. The answers to the questions are well known [12, 11, 38], but the determination of such a set is largely dependent on the final purpose of the application. It is, in many cases, an expensive computational procedure. In this investigation, the theoretical results that allow the enumeration of the cycles of a graph will be complemented with proposed algorithms to extract a set of cycles that are meaningful from the physical point of view of the GCS/SF problem.

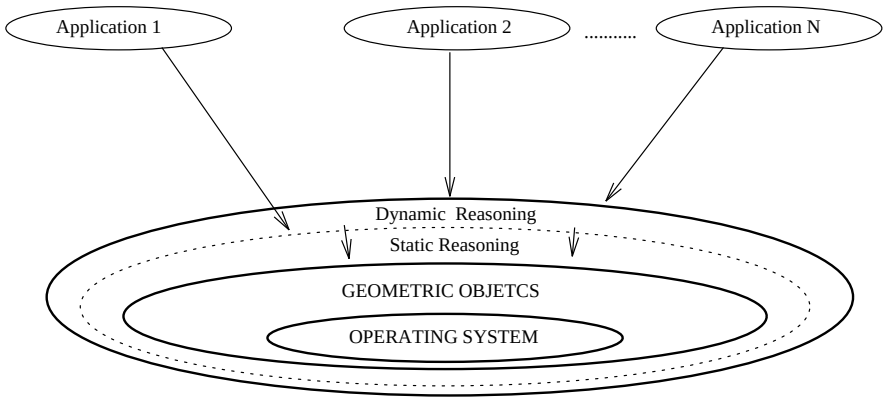

Figure 1.2 Centralized Geometry Server

1.4 Outline of the Proposed Research

The long term goal of this research is the achievement of a Dynamic and Static Reasoning Server applicable in CAD / CAM / CAPP environments. However, the degree to which knowledge in these areas (static/dynamic) has developed presents a great unbalance, with the dynamic reasoning part being in the very early stages of exploration. This situation is taken into consideration by setting different objectives in each area. For Static Reasoning a robust implementation and application is proposed. In the Dynamic Reasoning area, research in the theoretical foundations and algorithms is targeted, along with exploration of contributing fields, and applications in a variety of domains. Activities of software development are comparatively reduced in this area.

Figure 1.2 shows the underlying philosophy in the creation of the centralized Geometry Server aimed at in this investigation. From the point of view of the user (programs or humans), a layer (Geometric Objects) extends the capabilities of the computer, presenting geometric objects as primitive types, similar to floating or integer numbers, character strings, etc. The Geometric Reasoning Server is anchored on top of this extension layer. It presents the Static and Dynamic Reasoning division previously mentioned. The Geometric Reasoning module serves the requests of variety of client applications.

1.4.1 Proposed Research in Static Reasoning

The nature of the problems addressed implies that Static Reasoning utilities have a role both in serving external calls, and requests from the Dynamic Reasoning part. A set of client programs/users share the capabilities of the library. The proposed scheme eliminates the opportunity of code replication, concentrates specialized knowledge in one place, under a unique standard of data structures and interfaces, and yet it is built in such a way that it allows clients to extend the capabilities with their own constructions and/or routines.

1.4.2 Proposed Research in Dynamic Reasoning

Figure 1.3 presents a conceptual division of different aspects in the research regarding Dynamic Reasoning. Individual discussion of each follows:

1.4.2.1 Equation Representation of the GCS/SF Problem

Two alternatives can be used in modeling the GCS/SF problem with a set of polynomial equations: (i) using non-canonical variables, which emphasize the parameters and positions of the entities in the world; or (ii) using the canonical variables, which stress the degrees of freedom allowed to the entities by the constraints. Methodologies for the modeling of the problem using both alternatives will be explored, and their strengths/weaknesses identified.

1.4.2.2 Solution of the Equation Form of the GCS/SF Problem

Regardless of the method of representation (canonical / non-canonical) numerical and symbolic techniques can be used for the solution of (polynomial) equations. However, as discussed before, given the need for *counting* the degrees of freedom of the entities, the numerical methods are not sufficient. Therefore symbolic techniques will be used in this research.

1.4.2.3 GCS/SF Problem Partition

According to the description to the GCS/SF problem, its representation in the form of a Spatial Constraint graph simply requires the mapping of entities into nodes and relations into arcs. A partition of the SC graph is then needed for the systematic production of the Equations Form of the GCS/SF problem. The existing theory [33, 12, 11] will be used in order to produce a partition that also facilitates the application of preprocessing techniques proposed in this investigation, and discussed later.

Symbolic manipulation techniques [19, 36], based in look-up tables have the characteristic that they require a canonical representation of the constraints. Further, they have the limitation of working only with the topological part of trivial constraints. Geometry has to be dealt with separately, and non-trivial constraints are not contemplated. In

response to this limitation, in this work the calculation of a Grobner Basis for the polynomial representation of the GCS/SF problem will allow: (i) the modeling by canonical and non-canonical variables; (ii) the inclusion of non-trivial constraints; and (iii) the consideration of the geometrical and topological aspects of the problem. This application of Algebraic Geometry techniques requires that a direct relation be established between the Grobner Basis of the polynomial form of the GCS/SF problem and its solution space (the degrees of freedom of the scene). This relation is an important part of the contribution of this investigation.

1.4.2.4 Divide & Conquer Solution for the GCS/SF Problem

The partition of the GCS/SF problem was first used in topology oriented solution techniques in [19, 36]. The present work will use Divide & Conquer techniques in connection with Grobner Basis calculation as a means to make efficient use of characteristics of subproblems. Previous works in GCS/SF do not elaborate on the strategies of problem partition. This investigation considers SC graph partition as a step to produce the equation form of the GCS/SF problem. Beyond this consideration, the partitioning of the Spatial Constraint graph will allow the application of Divide & Conquer techniques. Therefore, two important aspects will be addressed: (i) methods for partitioning the problem; and (ii) usage of the solution of subproblems in the construction of the solution for the full GCS/SF scenario.

1.5 Dissertation Outline

This thesis is organized as follows:

Chapter 2 examines the structure of the Geometric Reasoning Server. Static Reasoning tasks have the dual purpose of (i) addressing geometric problems which are themselves relevant to the CAD / CAM / CAPP processes; and (ii) support of of the solutions to the GCS/SF (dynamic) problem.

Chapter 3 establishes a methodology for the formal expression of the GCS/SF problem in polynomial forms. Relevant background in Algebraic Geometry is discussed. Different physical situations of the GCS/SF are mapped into properties in the Algebraic Geometry domain, and vice versa. This chapter therefore gives theoretical support to the solution of the GCS/SF problem.

Chapter 4 describes an alternative methodology for the expression of the GCS/SF problem by using the subgroups of the Special Euclidean group $SE(3)$ (canonical formulation). This alternative representation allows for: (i) a more efficient representation of the GCS/SF problem; (ii) a direct relation between variables and physical degrees of freedom of the entities involved; and (iii) the application of automated reasoning in terms of degrees of freedom. This automated reasoning may be implemented in the form of look-up tables, rewriting rules, etc.

Chapter 5 studies partition techniques for the GCS/SF problem. The subdivision of the problem is required not only for a systematic enumeration of all the equations governing the scene, but for the application of Divide & Conquer techniques. Since these techniques can be implemented by partitioning the Spatial Constraint graph of the GCS/SF problem, this chapter studies the mathematical structure of graphs, and the representation of graphs as linear spaces. The partition of the SC graph is then related to the determination of a basis for particular subspaces of that linear space. After a theoretical background is set up, heuristically efficient algorithms for the partition of the GCS/SF problem are proposed. Chapter 5 concludes with an example of the application of the theory reviewed and algorithms developed.

Chapter 6 evaluates the proposed techniques for the solution of the GCS/SF problem in terms of the computer resources spent on them. These techniques have varying effectiveness depending on the characteristics of the problem at hand. These characteristics include: (i) Two body vs. multi-body systems; and (ii) trivial vs. non-trivial constraints. The choices for solution include: (a) *modeling* of the problem by canonical or non-canonical variables; and (b) *solving* the polynomial form of the GCS/SF prob-

lem by handling the whole set of polynomials, or by identifying and preprocessing local subproblems in order to contribute to the general solution.

Chapter 7 illustrates several applications of this research, in the areas of Mobility Analysis and Kinematic Analysis of Mechanisms. This chapter also discusses the interaction of the Geometric Reasoning server with a client program; in this case a Feature Extraction client module uses diverse libraries of the Static Geometric Reasoning server.

Chapter 8 establishes the limitations and potentials of the present investigation, and highlights the areas in which future research would be most fruitful.

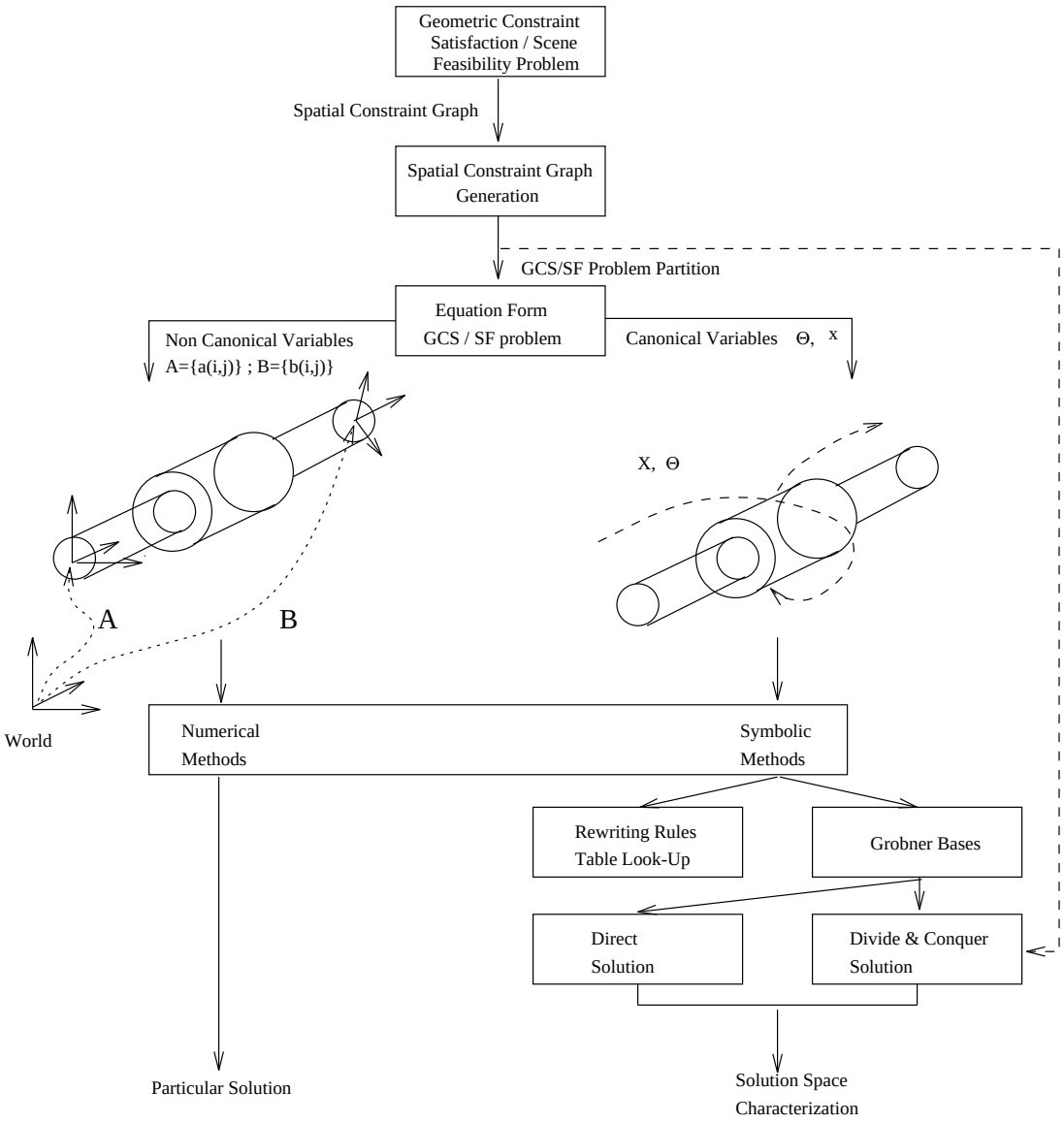

Figure 1.3 Constraint Satisfaction Problem. Methods of Solution

CHAPTER 2

A Static Geometric Reasoning Server

2.1 Introduction

In the previous chapter the rationale for the research and development of a server specialized in Geometric Reasoning was introduced. The basic division between *dynamic* and *static reasoning* was established, and an overview of the theoretical background behind these areas was presented. This chapter will discuss the *Static Reasoning* server designed and developed as part of this investigation. Besides the central Geometric Reasoning module, it contains complementary libraries which are essential to make it accessible by clients (humans or programs). A brief description of the supporting modules follows, and the conclusions of this development are drawn. A later chapter discusses the application of the Static Reasoning server and other modules in the domain of Feature Recognition.

In CAD / CAM / CAPP environments, apparently unrelated problems in many cases share an underlying theoretical area. Failure in factorizing and developing the specialized common knowledge inherent in the problems results in a series of separate and overlapping *ad hoc* attempts to solve them. Although successful in the immediate sense, these attempts produce large replication, and waste of efforts, coding, knowledge, etc. When another application that shares the same knowledge is encountered, the same situation and patterns repeat. To prevent this situation, a centralized server specialized in *geometry* has been designed and implemented.

2.2 Static Reasoning in the GCS/SF problem

The previous chapter discussed the need for a centralized Static Geometric Reasoning server. Besides the obvious applicability in CAD / CAM / CAPP, static reasoning serves a supporting role in solving dynamic reasoning problems. This section elaborates on this statement.

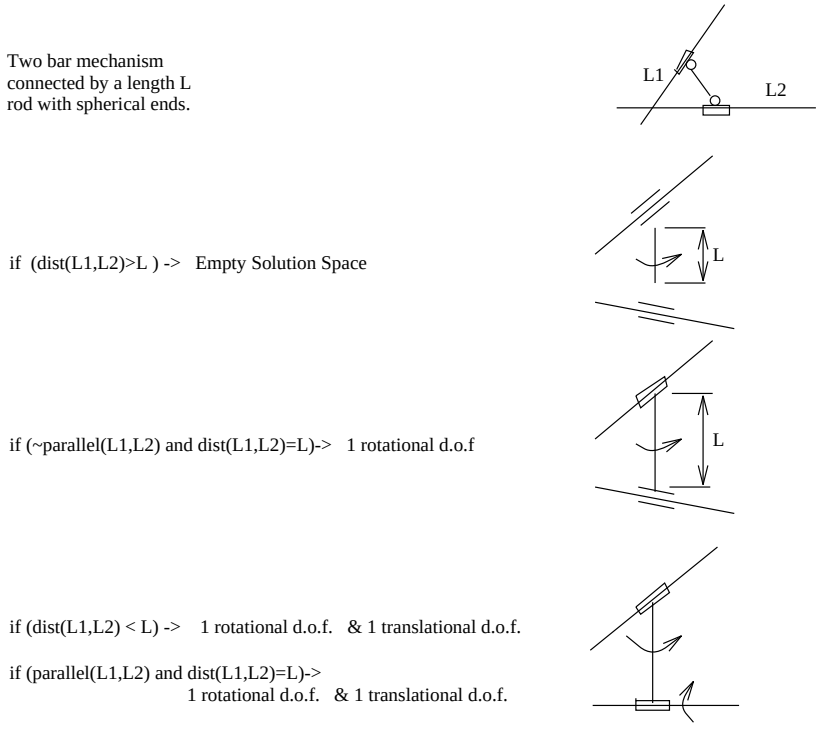

Figure 2.1 Static Reasoning Support for Dynamic Reasoning

The instance of the GCS/SF problem presented in Figure 2.1 belongs to the domain of kinematic analysis of mechanisms. Given a kinematic chain of entities (bodies) and constraints (joints) the goal is to determine how many, and which degrees of freedom the chain presents. The example presented involves two lines L_1 and L_2 in arbitrary position in the space, and a rigid bar, equipped with spherical joints in both extremes, which connects L_1 and L_2. As a crude approximation to the appropriate solution, a case-by-case analysis of the dimension of the solution space would originate the following algorithm:

```
function 3D_slider(L1, L2 : line; L : real) : {d.o.f.}
0 {
1               if (distance(L1, L2) > L) →
2                   return(Φ);
3               else if (distance(L1, L2) = L) →
4                   if (parallel(L1, L2)) →
5                       return({θ, x});
6                   else
7                       return({θ});
8                   fi
9               else if (distance(L1, L2) < L) →
10                  return({θ, x});
11              fi
12 }
```

This example, illustrates the fact that geometric conditions ($distance()$ and $parallel()$ in the example) have the capability to change the structure of the solution space, although the topological specification of the constraining relations remains the same. Therefore, the calculation of a solution space in the case of the GCS/SF problem cannot be accomplished by separating geometrical or topological terms. It follows that dynamic reasoning cannot be performed without the support of static Reasoning routines.

2.3 Organization of the Static Reasoning Library

The Spatial Reasoning library has been implemented following the structure shown in Fig 2.2. Although its central goal is to provide geometric reasoning services, its implementation cannot be carried out in isolation, and its services cannot be exploited externally if no supporting utilities are provided. In this section the different support modules are discussed.

2.3.1 The Solid Modeler

The solid modeler contains the basic database in which the connectivity and geometry of the entities populating the world are stored and maintained. In this development an Object Oriented, Boundary Representation solid modeler was used. Since solid modeling is not the purpose of this investigation, a commercial product (ACIS) was selected as the basic platform. Object Oriented was used as the programming paradigm, since in many cases the nature of the objects handled is not known *a priori*; therefore a flexible data model, such as the one provided by the Object Oriented methodology, is required.

2.3.2 Standard Interface

To avoid that applications become dependent on a specific solid modeler, an Application Interface Specification (AIS) standard is enforced. The AIS assumes the interaction of the application with a generic modeler, leaving open the possibility of *replacing* the modeler without changing the program which uses it. This goal is attainable as long as both the new modeler and the Geometric Reasoning module comply with the AIS specification.

2.3.3 Application Interface

The design of the Geometric Reasoning system is such that a client program or application is able to use the Geometric Reasoning module as a library. This scheme enlarges the capabilities of the solid modeler. The client program may chose to interact with the World Administrator if there is need for services such as set or list libraries, naming/renaming of objects, management of attributes and intermediate results, deletion, debugging, etc. The interaction between the application and AIS is also open, in case that direct services of the solid modeler are required. As a last resort, the client application could interact at the level of the solid modeler itself. In that case no compatibility with other solid modeling systems should be expected.

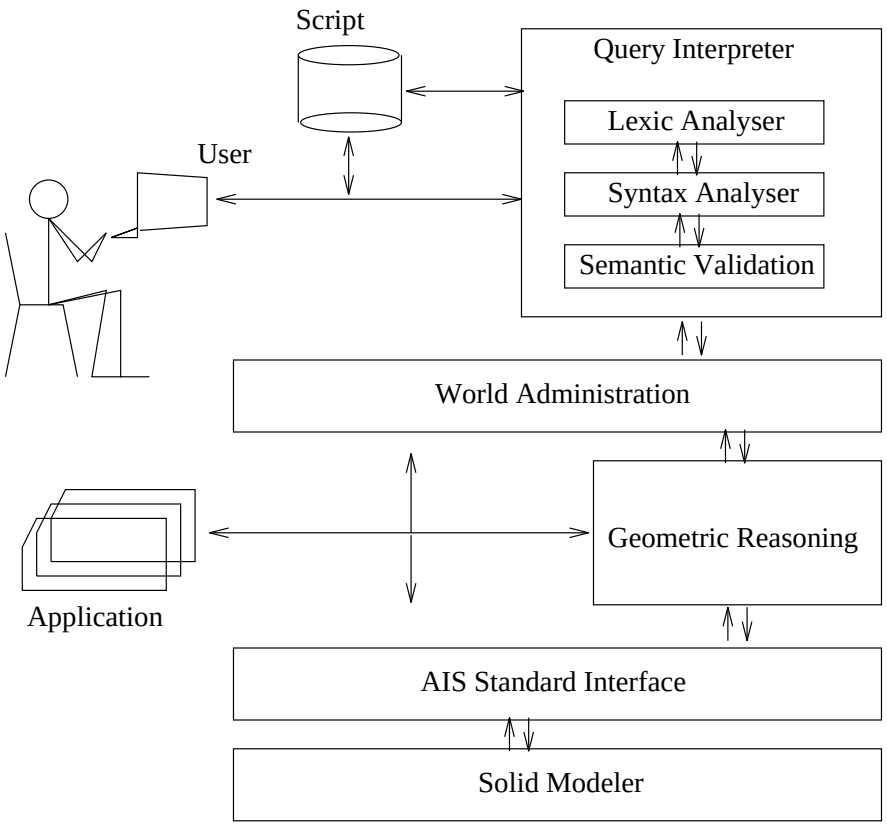

Figure 2.2 General Structure Geometric Reasoning Server

2.3.4 User and Script Interface

The goal of this interface is to allow the user to create, manipulate, display and query geometric objects *without creating a client application.* For this purpose, each routine that serves as entry point for a call from a program is also accessible to the user through the User Interface. In this way, an interactive user can access the services of the Geometric Reasoning server. It is also possible to make a log of a large sequence of commands, store the session and replay it for the purposes of demonstrations, or for corrections or modifications in the middle of a large chain of commands. Notice that a script can be also submitted by a client program, therefore providing another way of interaction. Consistent with the final purpose of the Geometric Reasoning Server, the language defined can be referred to as a *Geometric Prolog*, with capabilities for variable

instancing that are not present in pure Prolog. As usual in cases of language definition, the typical three submodules included are the Lexic Analyzer, Syntax Analyzer and Semantic Validator. Since these practical aspects are not central to this investigation, they will not be discussed here.

2.3.5 World Administration

This module is vital to any activity interacting with the objects kept in the solid modeling space, since the solid modeler does not provide database services. Therefore it is the responsibility of the World Administrator to perform the following tasks:

(1) *naming and attribute managing*, to guarantee the uniqueness of names for the objects in the world.

(2) *identification and extraction*, for extraction of selected components of objects.

(3) *object storage and retrieval*, for implementation of the data structures and classes which allow to organize, store and retrieve the entities in the world.

(4) *preservation of consistency* of the world upon object elimination.

2.4 Geometric Reasoning Module

From the point of view of software organization, the Geometric Reasoning module has the same status as other modules discussed above, such as the Solid Modeler, Application Interface, World Administrator, etc. However, from the point of view of its functionality and its importance for this investigation, it is singled out for separate discussion. In this section the domain and the functionality of the Geometric Reasoning module are presented.

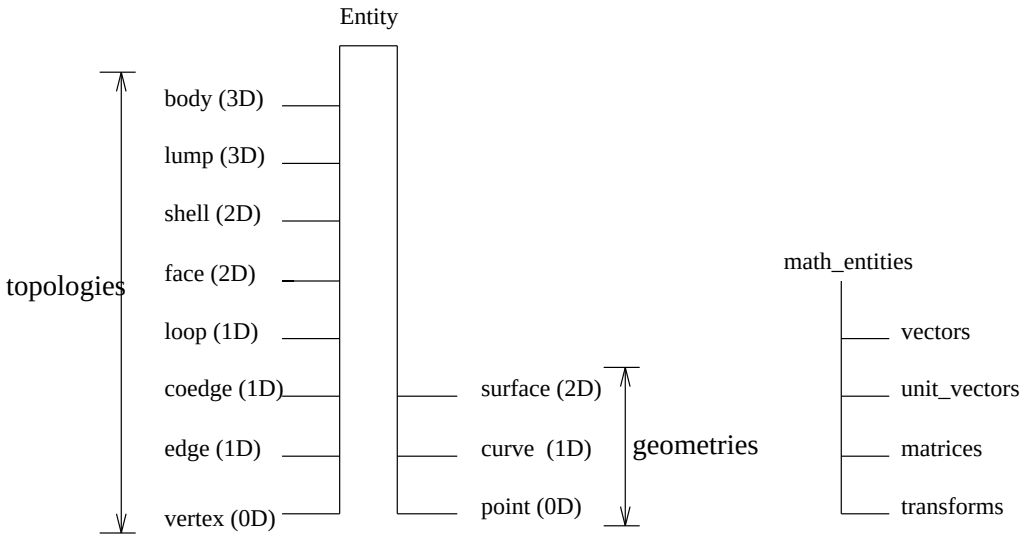

Figure 2.3 World Domain for the Geometric Reasoning Module

2.4.1 Domain and Hierarchies of Objects

The simplified structure of the objects in the World is shown in Figure 2.3. Given the functionalities of the module, there is a need to provide mathematical entities, which serve as arguments/results for many functions e.g. *vectors, unit vectors, matrices and transformations*. On the other hand, the geometric entities cover two main areas; (i) *topologies* (or topological entities), which convey the structure and connectivity of the elements in a given model; and (ii) *geometries*, which place the given topologies in the E^3 space, by giving them dimensions and positions.

In Figure 2.3 the dimensionalities of the different entities are shown. The figure also reflects the fact that as a general rule, a higher dimensionality entity uses for its definition lower dimensionality ones; for example a *COEDGE* uses *EDGE* and *VERTEX* for its definition. For the purpose of this investigation the world is assumed composed of flat $3D$ entities, including non-manifold objects. Therefore the *curve* and *surface* objects are restricted to *straight* lines and *plane* surfaces respectively.

2.4.2 Functionality

The functions developed as part of the Geometric Reasoning module have the following basic functionalities:

$$E \times E \times ... \times E \rightarrow \{TRUE, FALSE\} \tag{2.1}$$

$$E \times E \times ... \times E \rightarrow \{E\} \tag{2.2}$$

$$E \times E \times ... \times E \rightarrow [E] \tag{2.3}$$

$$E \times E \times ... \times E \rightarrow math\_entity \tag{2.4}$$

where E represents an entity. Line 2.1 represents *logical* or *boolean* queries. Lines 2.2 to 2.4 represent *constructive* queries. In line 2.2 the result of the function is a *set*, in which order is irrelevant, while line 2.3 involves a *list*, in which order is important. Line 2.4 introduces functions whose result is a mathematical entity (*vector*, *matrix*, etc).

Table 2.1 Reduced set of Logical Queries and Functionality

Function	*Arguments*	*Results*
colinear:	$E \times E$	*boolean*
coplanar:	$E \times E$	*boolean*
coincide:	$E \times E$	*boolean*
perpendicular:	$E \times E$	*boolean*
parallel:	$E \times E$	*boolean*
convex:	$E \times E$	*boolean*
concave	$E \times E$	*boolean*

Table 2.1 shows a classification of the logical functions according to main groups. Functions somehow overlap in their definition due to client program interface; for example *coincide(E1,E2)* tests whether two entities coincide. In the particular case of straight lines this function has to revert to *colinear(E1,E2)*. Therefore it is natural to design the system in such a way that *coincide()* be a generic function which is supported, in this given case, by *colinear()*. By providing the two functions, a seamless front end is presented to the user.

Table 2.2 Reduced set of Constructors and Functionality

Function	*Arguments*	*Results*
join_perpend:	$E \times E$	$point \times point$
distance:	$E \times E$	$real$
angle_between:	$E \times E$	$real \times real$
signed_angle_between:	$E \times E$	$real \times real$
intersection:	$E \times E$	$\{E\}$
sort_angular:	$\{point\} \times point \times vector$	$[point]$
projection	$E \times E \times vector$	$[E]$
planar_hull	$\{point\} \times vector$	$[point]$
space_hull	$\{point\}$	$shell$

Table 2.2 shows a classification of the constructive functions according to main groups. This table exemplifies the situation in which the exact result of a constructive operation is known only at execution time. For example, the projection of a line onto a jagged face in general would be a *set* of segments; a point if the line is perpendicular to the face; or a null entity if the projection falls outside the face limits.

Figures 2.4 and 2.5 show a partial view of the library structure. Figure 2.4 shows functions whose primary domain are geometric entities. Figure 2.5, corresponds to topological entities of dimension 1 and 2; EDGE and FACE.

2.4.3 Hierarchy

The structure of the library is such that many of the functions provided are themselves users of lower level utilities in the library kernel. Figure 2.6 shows an example of this situation. In this case, *space_hull*() calls an algorithm for generating the convex hull of a set of points in E^3 ($gift\_wrapping()$); this algorithm uses utility routines to calculate the angle between planes, ($signed\_angle\_between()$), extract coplanar points from a given set ($extract\_coplanar()$), and a planar convex hull function ($graham\_scan()$)based on the Graham Scan algorithm. The Graham Scan uses routines to sort a set of (coplanar) points based on the angle about a central point ($sort\_angular()$). The angular sorting

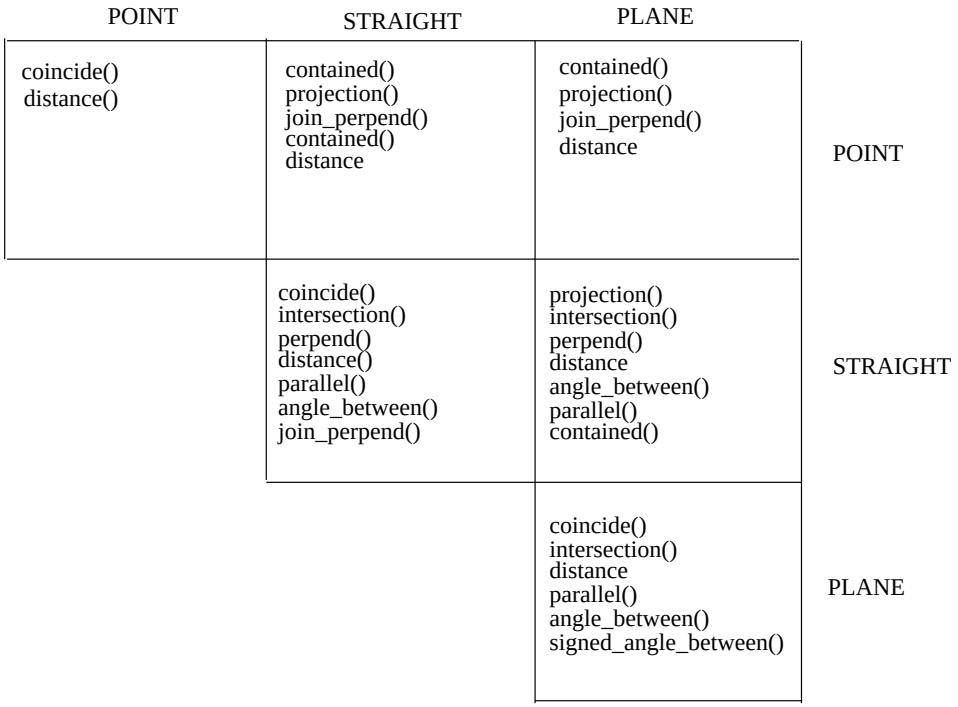

Figure 2.4 Reduced Library Structure of calls for Geometries

algorithm itself uses another routine to calculate the normal vector based on the boundary of a (planar) polygon ($calculate\_normal()$).

2.5 Summary

This chapter has discussed the motivation, organization and scope of a Static Geometric Reasoning server. This server presents applications in geometry-intensive operations in CAD / CAM / CAPP environments. It also supports the Dynamic Reasoning server, which addresses the the GCS/SF problem. The structure and scope of the constitutive parts has been presented. The following characteristics have been imparted to this server: (i) enlargement of the capabilities of the virtual machine to handle geometric objects; (ii) presentation of a compatible interface to supporting and supported software; (iii) interaction at level of library, direct user manipulation and script interpretation; (iv) flexible functionality, required because the influence of geometry on the dimensionality

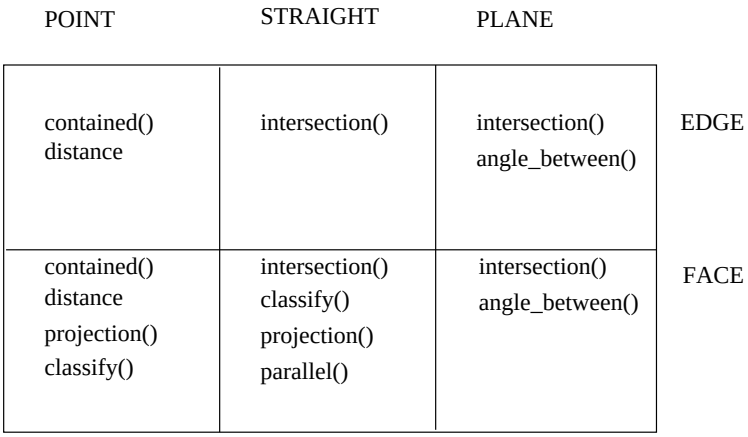

Figure 2.5 Reduced Library Structure of calls for Topologies

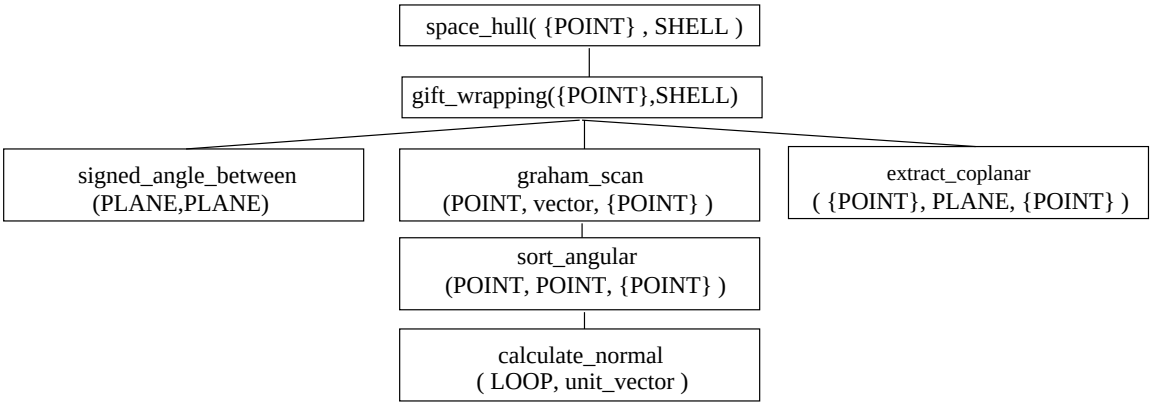

Figure 2.6 Example of hierarchical calls to solve the problem of 3D space convex hull

of the solution space; and (v) layered structure allowing extensibility to more complex services.

In a later chapter an application of static reasoning is presented. Such application, in which a client program for feature extraction uses geometric and database routines, illustrates the capabilities and interaction provided by the Geometric Reasoning server.

CHAPTER 3

Algebraic Geometry Solution to the GCS/SF Problem

In previous chapters the rational for a Geometric Reasoning server supporting CAD / CAM / CAPP activities was presented. The organization of a Static Geometric Reasoning module was then discussed. In this chapter the mathematical formulation of the GCS/SF problem (Dynamic Reasoning) will be presented. An important portion of this research involves the reasoning about the structure of the solution space, for the problem. Therefore, properties of the algebraic geometry methods used, relevant to the characterization of the solution space will be introduced. Based on these properties, an algorithm for constraint management will be proposed and illustrated with an example.

3.1 Polynomial Model for the GCS/SF Problem

This section demonstrates a methodology for stating the GCS/SF problem in terms of sets of polynomials. This methodology uses the (unknown) position frame of the entities in the scene. Therefore, in this case the unknowns of the GCS/SF problem are the elements of the matrix representing the frame. In this work, variables that result from such a formulation are called *non-canonical*, in contrast with *canonical* ones, discussed in following chapters.

The following conventions will be held:

Table 3.1 Elementary 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	$PLN=(p_2,n_2)$	$(p_1-p_2)\cdot n_2 = 0$
$LN-ON-LN$	$LN=(p_1,v_1)$	$LN=(p_2,v_2)$	$v_1\times v_2 = 0$ $(p_1-p_2)\times v_2 = 0$
$LN-ON-PLN$	$LN=(p_1,v_1)$	$PLN=(p_2,n_2)$	$(p_1-p_2)\cdot n_2 = 0$ $v_1\cdot n_2 = 0$
$PLN-ON-PLN$	$PLN=(p_1,n_1)$	$PLN=(p_2,n_2)$	$(p_1-p_2)\cdot n_2 = 0$ $n_1\cdot n_2 = \pm 1$

- *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 geometrical 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 relative position of entity i inside body (frame) j.

- C_i represents relations or constraints between entities. These relations are shown in the first column of Table 3.1.

- D_i represents displacements applied on the frames of the entities F_{ij}.

Let the configuration of entities be as shown in Figure 3.1, where B_2 can be assumed stationary with no loss of generality. Frames F_{i1} represent the position of distinguished element i ($i = 1..3$ in this case) of (and with respect to) body B_1. Similar statements can be made about F_{i2} with respect to body B_2. The goal is to find a position of B_1 which satisfies the relations $F_{i1} - R_i - F_{i2}$. For example, it may required that *point* F_{11} *be ON plane* F_{12}. That means, $F_{11} - ON - F_{12}$.

The procedure for modeling the problem in terms of sets of polynomials is:

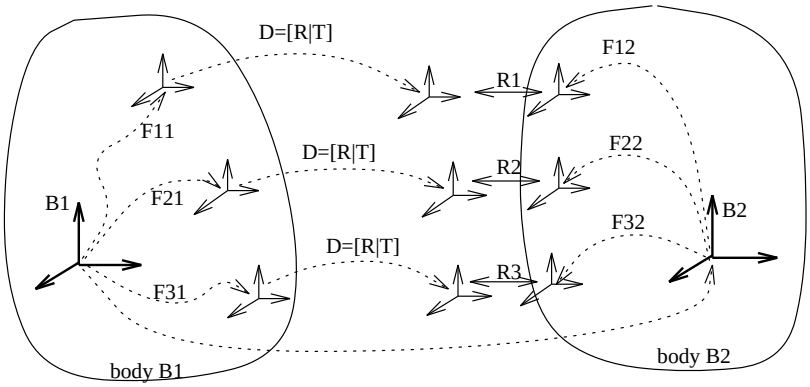

Figure 3.1 Methodology for Statement of Non-canonical Form of Scene Feasibility Problem

(1) Assume a (unknown) displacement $D = \begin{bmatrix} Rot & T \\ 0 & 1 \end{bmatrix}$ which will place body B_1 in the desired final position.

(2) Transform each entity to its new position: $F_{ij}.D$.

(3) Use Table 3.1 to model the proposed relations using the transformed entities. The proposed equations are not a minimal set since some redundant equations are produced; for example, $P - ON - LN$ can be expressed in two equations instead of three.

(4) Each relation (or constraint) produces a series of equations of the form $C_i(F_{i1}, D, B_2, F_{i2}) = 0$, which involves the corresponding entities F_{i1}, F_{i2}, the positions of the bodies D, B_2, and the particular form of the relation C_i:

$$C_1(F_{11}, D, B_2, F_{12}) = 0;\ C_2(F_{21}, D, B_2, F_{22}) = 0;\ C_3(F_{31}, D, B_2, F_{32}) = 0 \quad (3.1)$$

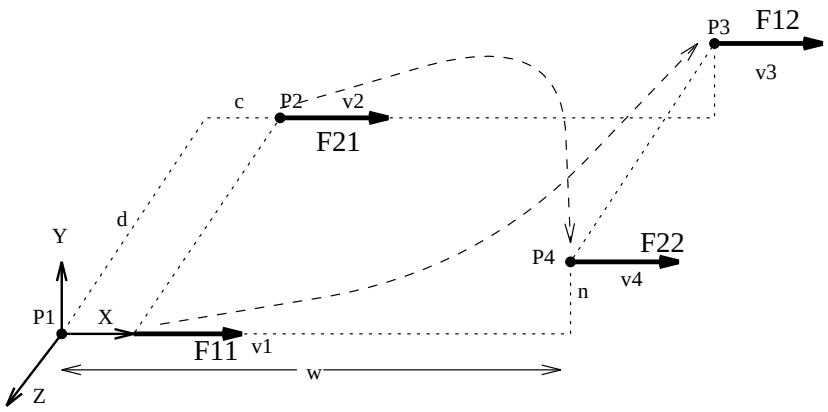

Figure 3.2 Simultaneous Line-to-Line Restriction between Pairs of Lines

3.1.1 Example. Constraint Expression

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 3.2) expressed parametrically, and assumed to be rigidly linked to each other. 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 is to find out 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.

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.

$$Rot = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}; T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3.2}$$

The transformed entities are

$$P_1' = T + Rot.P_1; \quad v_1' = Rot.v_1; \quad P_2' = T + Rot.P_2; \quad v_2' = Rot.v_2 \tag{3.3}$$

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

$$\begin{aligned} (P_1' - P_3) \times v_3 = 0; \quad & P_1' \in LN_3 \\ v_1' \times v_3 = 0; \quad & v_1' \parallel v_3 \\ (P_2' - P_4) \times v_4 = 0; \quad & P_2' \in LN_4 \\ v_2' \times v_4 = 0; \quad & v_4' \parallel v_4 \\ det(Rot) = +1; & \end{aligned} \tag{3.4}$$

The condition $det(Rot) = +1$ imposes dexterous orthonormality to the matrix $Rot = [v_1\ v_2\ v_3]$. Orthonormality implies $\|v_i\| = 1, (i = 1..3); \quad v_j.v_i = 0, (i \neq j)$. Dexterity implies $v_1 \times v_2 = v_3$. The corresponding polynomials are presented and discussed in later sections (equations 3.5).

The equations arrived at are polynomials, whose solutions determine the matrix D, and therefore the position of the (frame of) body B_1. Having established the expression of constraints in terms of polynomials, techniques for characterizing the solution for such a set are explored in following sections.

3.2 Grobner Basis and the GCS/SF Problem

In what follows, an introduction to an algebraic geometry technique called Grobner Basis construction will be attempted. Only those issues relevant to the GCS/SF problem are discussed. The interested reader is directed to [21, 7] for details. The following is some relevant terminology.

K[x1,x2,...,xn]: ring of n-varied polynomials over the coefficient field K.

Algebraic Closure : The algebraic closure of a field K, $\bar{K}$, is the field of all roots of all polynomials in $K[x_1, x_2, ..., x_n]$. If K is R the set of real numbers, then $\bar{K}$ is the set of complex numbers C.

Ideal of F : The Ideal of a polynomial set $F = \{f_1, f_2, f_3, ..., f_n\}$ is:

$I_{K[x_1,x_2,...,x_n]}\langle F\rangle = \{g_1.f_1 + g_2.f_2 + ... + g_n.f_n | g_i \in K[x_1, x_2, ..., x_n]\}$.

The notation is usually simplified to: $I\langle F\rangle$. One says that F is a *basis* for $I\langle F\rangle$.

Radical(F) : $\{f | \exists k \; s.t. \; f^k \in Ideal(F)\}$

Algebraic set V(I) : Given an ideal $I \subset K[x_1, x_2, ..., x_n]$ generated by the set $F = \{f_1, f_2, f_3, ..., f_m\}$, its algebraic set $V(I)$ is defined by:

$V(I) = \{x \in \bar{K}^n | f(x) = 0, \forall f \in I\}$; therefore, $(f_i(x) = 0 \; \forall f_i \in F) \rightarrow (x \in V(I))$

Zero Dimension : An Ideal I is *zero-dimensional* if $V(I)$ is finite.

Ordering : the set of variables $\{x_1, x_2, ..., x_n\}$ is totally ordered under a given order $\prec$ if $\forall x_i \neq x_j$, either $x_i \prec x_j$ or $x_j \prec x_i$.

Lexicographic Order $\prec_l$: Given two terms $t1 = x_1^{\alpha_1}.x_2^{\alpha_2}..x_n^{\alpha_n}$ and $t2 = x_1^{\beta_1}.x_2^{\beta_2}..x_n^{\beta_n}$, then $t1 \prec_l t2$ iff $\exists i \leq n$ such that $\alpha_j = \beta_j$ for $i \prec j \preceq n$ and $\alpha_i < \beta_i$.

Degree : $deg(t) = deg(x_1^{\alpha_1}.x_2^{\alpha_2}..x_n^{\alpha_n}) = \alpha_1 + \alpha_2 + ... + \alpha_n$

Degree Order $\prec_d$: $t1 \prec_d t2$ iff $deg(t1) < deg(t2)$ or $deg(t1) = deg(t2)$ and $t1 \prec_l t2$

head(f), ldcf(f) : For a given order, and a given ring $K[x_1, x_2, ..., x_n]$, $head(f)$ is the largest (in the sense of $\prec$) term in polynomial f. $ldcf(f)$, the *leading coefficient* of f, is the coefficient of $head(f)$ in f. Therefore $f = \underline{ldcf(f).head(f)} + tail(f)$.

Normal Form : Given $F = \{f_1, f_2, f_3, ..., f_n\}$ and p where $F \subset K[x_1, x_2, ..., x_n]$ and $p \in K[x_1, x_2, ..., x_n]$, there exists a decomposition of p :

$p = NF(F, p) + \sum_{f_i \in F}(\alpha_{f_i}.f_i)$ (with $\alpha_{f_i} \in K[x_1, x_2, ..., x_n]$) such that $NF(F, p)$ cannot be further decomposed as $\sum_{f_i \in F}(\beta_{f_i}.f_i)$ with $\beta_{f_i} \in K[x_1, x_2, ..., x_n]$. The term $NF(F, p)$ is called a *normal form of p with respect to F* and $NF(F, p)$ is a

residual of the *reduction* of p *with respect to* F. The reduction process is denoted as $p \longrightarrow_F NF(F,p)$.

Grobner Basis : A Grobner Basis $GB \subset K[x_1, x_2, ..., x_n]$ is a set of polynomials such that $NF(GB, f)$ for every f is unique; it does not depend on the *sequence* of reduction of f with respect to GB. Therefore, $f \longrightarrow_{GB} p_1$ and $f \longrightarrow_{GB} p_2$ imply $p_1 = p_2$ (the converse is not true). Also, if $NF(GB, f) = 0$ then $f \in I\langle GB\rangle$. If $NF(GB, f) \neq 0$ it implies some of the common roots of F are not roots of f; therefore the set of roots common to F and f is more restricted than the set of roots of F.

Reduced Grobner Basis : A Grobner Basis $GB = \{g_1, ..., g_n\}$ is a *Reduced Grobner Basis* if:

(1) for all $f_i \in GB \quad ldcf(f_i) = 1$

(2) for all $f_i \in GB \quad NF(GB - \{f_i\}, f_i) = f_i$

Let $F = \{f_1, f_2, f_3, ..., f_n\}$ be a polynomial set in $K[x_1, x_2, ..., x_n]$, and $I\langle F\rangle$ be its ideal. If another set $G = \{g_1, g_2, g_3, ..., g_n\}$ is *basis* of $I\langle F\rangle$ then, every root of F is also root of G, and conversely.

Given a polynomial $f \in K[x_1, x_2, ..., x_n]$ one may want to eliminate a term t of f with the help of another polynomial $g \in K[x_1, x_2, ..., x_n]$ by multiplying the $head(g)$ by some term such that on subtracting the result from f, t disappears. For this to happen, it is necessary that $g \prec f$. It is said then that f *is reduced with respect to* g. It is written as $f \xrightarrow{g} h$, where h is the result of the subtraction. In the process of iterated reductions with respect to elements of $K[x_1, x_2, ..., x_n]$, the position of the h's in the ordering $\prec$ decays. One of two things may occur: either f reduces to 0, or all the remaining g's are *bigger* than the final h, and therefore f cannot be further reduced. The last product of the reduction process is a *normal form of* f *with respect to* $K[x_1, x_2, ..., x_n]$, $NF(K[x_1, x_2, ..., x_n], f)$. In the described process, different sequences of reduction are possible, and they do not, in general, produce the same $NF(K[x_1, x_2, ..., x_n], f)$ result. If a set of polynomials F is

used for the decomposition, $NF(F, f)$ can be considered as the part of f that cannot be expressed as a combination of the polynomials $f_i \in F$.

Several additional comments are pertinent at this point:

- Grobner Basis forces $NF(GB, F)$ to be unique, thus providing a way to examine whether an arbitrary polynomial p is in $I\langle F\rangle$ or not. If $p \in I\langle F\rangle$ then $NF(GB, p) = 0$. Otherwise, it represents an independent polynomial. Intuitively, Grobner Basis behaves in a manner analogous to a vector basis in linear spaces: if a vector can be expressed as a linear combination of the base vectors, it is in the space. In that case, any common root for the polynomials in the basis also makes any linear combination of them to vanish.

- In a property described later (triangularity of elimination ideals), Grobner Basis presents a characteristic similar to triangulation of a matrix A in solving a linear system $A.x = b$. A triangular form allows the incremental determination of the solution point.

- In a Reduced Grobner Basis there is no redundancy in the polynomials present, since each polynomial is equal to its normal form with respect to the remaining ones. The value of this property in the solution of the polynomial system is that it reduces to a minimum the polynomials to be manipulated and/or solved.

- An algorithm to calculate the Grobner Basis $GB(F)$ for a polynomial ideal $I\langle F\rangle$ is provided by Buchberger in [7]. Several implementations are available in packages such as Mathematica, Maple, Macaulay, etc. The condition for termination of the Buchberger's algorithm relies heavily on the fact that a total order on $K[x_1, x_2, ..., x_n]$ [7, 21] can be defined. Since a *decreasing* sequence (in the sense of $\prec$) of terms is finite, a reduction process of a polynomial p with respect to a set F is bound to stop.

In the next sections, the theoretical basis developed here will be used to exploit the properties of Grobner Basis in the solution of the GCS/SF problem.

3.2.1 Algebraic Geometry and the GCS/SF Problem

The GCS/SF problem takes place in a world W, with a set of relations R. If a set of entities $S = \{e_1, ..e_n\}$ responds to the constraints, it is said that S *is feasible for* W *and* R , and this fact is written as $S = feasible(W, R)$. If the polynomial form of the problem is $F = \{f_1, f_2, ..., f_n\}$, it is said that $F = poly\_form(W, R)$. Since S is a solution for F, it is denoted as $S = solution(F)$.

Given F ($F = poly\_form(W, R)$ and $S = feasible(W, R)$), there is an associated ideal $I\langle F\rangle$. For any polynomial set F, the Grobner Basis $GB(F)$ is an alternative set, which generates the same ideal $I\langle F\rangle$, but has important properties in characterizing the solution space and producing solution points.

The following are some of the properties of Grobner Basis:

(1) $I\langle GB(F)\rangle = I\langle F\rangle$.

(2) F is solvable in $\bar{K}$ iff $1 \notin GB(F)$.

(3) Given a lexicographic order $x_1 \prec x_2 \prec ... \prec x_n$ $\forall i, s.t. 1 \leq i \leq n$, we have: $GB(F) \cap K[x_1, x_2, ..., x_i]$ is a (reduced) Grobner Basis for the elimination Ideal $I_{K[x_1,x_2,...,x_n]}\langle F\rangle \cap K[x_1, x_2, ..., x_i]$.

This property establishes that GB(F) is triangular set; 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) If G is the reduced Grobner Basis for an Ideal $I \in K[x_1, x_2, ..., x_n]$, I is *zero-dimensional* iff $\forall x_i \in \{x_1, x_2, ..., x_n\}$, G contains a polynomial whose head term is a pure power of x_i, i.e. of the form x_i^d for some integer d.

This property allows one to determine, by inspection, whether the set of polynomials has finitely or infinitely many solutions.

(5) The Grobner basis $G1$ for a zero-dimensional ideal I based on the order $\prec_m$ can be converted into another basis $G2$ under another ordering $\prec_l$.

This property allows one to compute *total degree* Grobner Bases for certain purposes, and only when it is required, to transform them into *lexicographic* Grobner Bases (computationally more expensive), provided that they correspond to a zero-dimensional ideal.

(6) $\forall f \in K[x_1, x_2, ..., x_n], \forall y \notin \{x_1, x_2, ..., x_n\}$
$f \in Radical(F) \Leftrightarrow (1 \in GB(F \cup \{y.f - 1\}))$
The equation $y.f - 1 = 0$ ensures $f \neq 0$. Therefore, this property establishes that f presents the same zeros as F iff the system $F \cup \{y.f - 1\}$ is inconsistent, i.e. it is impossible for f not to be zero when F is.

These properties translate into propositions about the solvability and characteristics of the solution for the GCS/SF problem. Some of the consequences of the properties follow:

(1) **Proposition 1**
$S = solution(F)$ iff $S = solution(GB(F))$.
This is a consequence of the fact that F and $GB(F)$ span the same polynomial ideal (Property 1). In the context of the GCS/SF problem, a set of polynomials representing constraints is indirectly analyzed by calculating the Grobner Basis of its polynomial ideal and solving it by using the properties discussed below.

(2) **Proposition 2**
$1 \in GB(F) \Rightarrow S = solution(F) = \Phi$
Property 2 above establishes that if the field is algebraically closed, finding "1" or a constant polynomial in GB(F) implies the equation "0=1" leading to the fact that F has no solution in that field. However, the converse proposition has to be carefully used:
If $1 \notin GB(F)$, a solution exists, although it might be complex. Therefore, an additional check on the results of a numerical algorithm to ensure a *real* solution is needed.

(3) **Proposition 3**

If $I\langle F\rangle$ is a zero-dimensional ideal, then the set F (and $GB(F)$) has a finite number of solutions. Therefore $S = feasible(W, R)$ has a finite number of configurations. The zero-dimensionality of I can be assessed by applying property 4 above.

(4) **Proposition 4**

Let a new constraint be represented by polynomial f. Then:

f is redundant to F iff $(1 \in GB(F \cup \{y.f - 1\}))$ for a new variable y.

Property 6 above helps to determine whether an additional constraint is redundant by examining if the satisfaction of the new, additional constraint is unavoidable when the initial set of constraints is satisfied. An alternative test can be implemented by recalling that a polynomial f is redundant if its normal form $NF(GB(F), f) = NF(F, f)$ is equal to zero.

These properties and propositions provide a theoretical framework for the solution of the GCS/SF problem. The construction of an algorithm using these facts will be discussed in following sections.

3.2.2 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 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), consistency (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 lexicographic order is extracted (Property 5) and solved (Property 3). 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. In the algorithm presented below, the propositions or properties relevant to some important instructions are displayed at the left hand side:

		{Pre: W a fixed scenario }
	1	$F = \{\}$
	2	$GB_t = \{\}$
	3	do new relation R_i
	4	{Inv: F is consistent, non-redundant, non-zero-dimensional }
	5	$R = R \cup \{R_i\}$
	6	$f = poly\_form(W, R_i)$
Proposition 2	7	if $(1 \in GB_t(F \cup \{f\}))$ then
	8	stop (system is inconsistent)
	9	else
Proposition 4	10	if $(f \in Radical(F))$ then
	11	skip (f is redundant)
	12	else
	13	$F = F \cup \{f\}$
Proposition 1	14	$GB_t = GrobnerBasis(F, \prec_t)$
Proposition 3	15	if ($ZeroDimension(GB_t)$) then
	16	break loop
	17	else
	18	skip (next relation-constraint)
	19	fi
	20	fi
	21	fi
	22	od
Property 5	23	$GB_l = GrobnerBasis(F, \prec_l)$
Property 3	24	$S = triangular\_solution(GB_l)$
		{Post:$R = \{Ri\}$ a set of relations; $S = feasible(W, R)$ }

The limitations of Grobner Basis (and for that matter, any symbolic algebraic geometry method solving this problem) is the explosive computational complexity of the method, and its still-unexplored behavior in dealing with floating point (real) arithmetic.

If F is a set in $K[x_1, x_2, ..., x_n]$, with maximum exponent m, the Grobner Basis can contain polynomials of degree proportional to 2^{2^m} [21].

3.2.3 Example 1. Grobner Basis for the Constraint Set

This section continues the example described previously, which demonstrated how to build the polynomial formulation of the constraint set. The reader might want to refer back to Figure 3.2 for details on the example.

The basic condition of (dexterous) orthonormality of the D matrix produces the following set of equations:

$$\begin{gathered}
{x_{11}}^2 + {x_{21}}^2 + {x_{31}}^2 - 1 = 0, \quad {x_{12}}^2 + {x_{22}}^2 + {x_{32}}^2 - 1 = 0 \\
{x_{13}}^2 + {x_{23}}^2 + {x_{33}}^2 - 1 = 0, \quad x_{13}\,x_{12} + x_{23}\,x_{22} + x_{33}\,x_{32} = 0 \\
x_{11}\,x_{12} + x_{21}\,x_{22} + x_{31}\,x_{32} = 0, \quad x_{21}\,x_{32} - x_{31}\,x_{22} - x_{13} = 0 \\
x_{31}\,x_{12} - x_{11}\,x_{32} - x_{23} = 0, \quad x_{11}\,x_{22} - x_{21}\,x_{12} - x_{33} = 0
\end{gathered} \tag{3.5}$$

The lexicographic order used in this example, for the calculation of the Grobner Basis is:

$$x_{11} \succ x_{12} \succ x_{13} \succ x_{21} \succ x_{22} \succ x_{23} \succ x_{31} \succ x_{32} \succ x_{33} \succ T_x \succ T_y \succ T_z \tag{3.6}$$

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

$$(P_1' - P_3) \times v_3 = 0 \quad (P_1' \in LN_3); \quad v_1' \times v_3 = 0 \quad (v_1' \parallel v_3) \tag{3.7}$$

produce an initial polynomial set:

$$\{d + T_z = 0, n - T_y = 0, x_{31} = 0, -x_{21} = 0\} \tag{3.8}$$

The Grobner Basis corresponding to this condition is shown below. The $\underline{f}$ notation is used to stress the fact that f is $head()$ of a polynomial:

$$\underline{T_z} + d = 0 \tag{3.9}$$

$$\underline{T_y} - n = 0$$

$$\underline{x_{32}}^2 + {x_{33}}^2 - 1 = 0$$

$$\underline{x_{31}} = 0$$

$$\underline{x_{23}}^2 + {x_{33}}^2 - 1 = 0$$

$$\underline{-x_{22}} + x_{22}\,{x_{33}}^2 - x_{23}\,x_{32}\,x_{33} = 0$$

$$\underline{x_{22}\,x_{32}} + x_{23}\,x_{33} = 0$$

$$\underline{x_{22}\,x_{23}} + x_{33}\,x_{32} = 0$$

$$\underline{x_{22}}^2 - {x_{33}}^2 = 0$$

$$\underline{x_{21}} = 0$$

$$\underline{x_{13}} = 0$$

$$\underline{x_{12}} = 0$$

$$\underline{x_{11}} + x_{23}\,x_{32} - x_{22}\,x_{33} = 0$$

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, indicates that inconsistency of the constraint with the preexisting scene cannot be concluded. Yet, it is possible to have a solution with complex variables which is not physically realizable. Second, Proposition 3 and Property 4 indicate a multi-dimensional ideal and consequently an infinite number of solution configurations. Any claim to a zero-dimensional ideal, and hence to a completely determined configuration for the scene, is trivially discarded by the fact that T_x does not appear in any polynomial in the basis (therefore it cannot be $head()$ of any polynomial, Property 4). Further inspection of the basis also indicates that x_{33} does not appear in

any of the head terms. These facts indicate a two dimensional ideal, because T_x and x_{33} can be given arbitrary values and still a real solution would exist for the system. This conclusion is consistent with the physical fact that the constraint would still be valid under arbitrary rotations around the line LN_3 and translations along it. While the translational degree of freedom is easily related to T_x, the relationship of the rotation to x_{33} is less intuitive.

Suppose the second constraint $(LN_2 - ON - LN_4)$ is added, resulting in the equations

$$(P_2' - P_4) \times v_4 = 0 \quad (P_2' \in LN_4); \quad v_2' \times v_4 = 0 \quad (v_2' \parallel v_4) \tag{3.10}$$

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

$$\begin{gathered}
\underline{T_z} + d = 0 \\
\underline{T_y} - n = 0 \\
\underline{x_{33}} + 1 = 0 \\
\underline{x_{32}} = 0 \\
\underline{x_{31}} = 0 \\
\underline{x_{23}} = 0 \\
\underline{x_{22}}^2 - 1 = 0 \\
\underline{x_{21}} = 0 \\
\underline{x_{13}} = 0 \\
\underline{x_{12}} = 0 \\
\underline{x_{11}} + x_{22} = 0
\end{gathered} \tag{3.11}$$

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 for the accumulated set of equations has each variable in the head term of an equation in the basis and would therefore be 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:

$$\begin{gathered}
\underline{T_z} + d = 0 \\
\underline{T_y} - n = 0 \\
\underline{T_x}^2 - c^2 + w^2 - 2\,w.T_x = 0 \\
\underline{x_{33}} + 1 = 0 \\
\underline{x_{32}} = 0 \\
\underline{x_{31}} = 0 \\
\underline{x_{23}} = 0 \\
\underline{c.x_{22}} + w - T_x = 0 \\
\underline{x_{21}} = 0 \\
\underline{x_{13}} = 0 \\
\underline{x_{12}} = 0 \\
\underline{x_{11}}\,c - w + T_x = 0
\end{gathered} \tag{3.12}$$

If yet one more condition is set, unless it is redundant, the system becomes inconsistent; for example, the requirement $P_2 - ON - P_4$, the Grobner Basis produces an inconsistency. In this case it is a *topological* inconsistency, which will in general impede the solution, *except* for a special set of values, i.e. for a very special point on the line L_4 (which is not P_4) to receive the point P_2. The inconsistent Grobner Basis in this case would be:

$$GB = \{1\} \tag{3.13}$$

The four instances of $GB(F)$, sequentially calculated as constraints are added, demonstrate how the GCS/SF problems might be solved. Further, it can be noticed that the $GB(F)$ can be ordered by using the prescribed term ordering. For example, equations 3.12 allows the variables to be solved in the order $T_z, T_x, ..., x_{11}$. The equations themselves are in triangular form. It can be seen that x_{11}, which is highest in the order, appears in only one equation while T_x, which is lower, appears in a number of equations.

3.3 Summary

In this chapter the problem of reasoning about geometric constraints was addressed using Grobner Bases. The Grobner Basis of a polynomial set $F = \{p_1, p_2, ..., p_n\}$ has several properties for characterization of the variety of the polynomial ideal. From the GCS/SF problem perspective, these properties allow to determine: (i) if there are remaining spatial degrees of freedom among the entities in a given scenario or world; (ii) the redundancy of a constraint in the context of a set of constraints; and (iii) the (in)consistency of the set of constraints F. An algorithmic explanation of how the Grobner Basis properties can be exploited in a constraint based design/planning situation was given and discussed by means of an example.

The theoretical underpinnings of Grobner Basis have been found to be useful in giving mathematical expression to different actual situations concerning the GCS/SF problem, such as finite or infinite number of configurations (including no possible configuration) corresponding to inconsistent constraint sets, as well as a formal definition and detection procedure for redundant constrains. On the other hand, Grobner Basis provides a framework for the integral treatment of topological and geometrical consistency in the set of constraints.

Although the example described above is relatively simple (12 variables and 20 equations), given the high computational complexity of the Grobner Basis construction it illustrates a critical limitation of the formulation discussed. Emphasis therefore needs to be placed on reducing the size of the problem to a minimum. Also, the difficulty in

the physical interpretation of the degrees of freedom was manifest in the solutions found. A more "natural" set of variables which addresses these problems is therefore required. These problems are addressed in following chapters.

CHAPTER 4

Group Based Solution for the GCS/SF Problem

In previous chapters, the GCS/SF problem was posed as a system of polynomial equations and its Grobner Basis was used to characterize their solution space. The properties of the Grobner Basis allow the sequential addition of constraints between entities in a scenario. These properties permit the response to questions about the dimensionality of the solution space or, in physical terms, the multiplicity of the feasible scenarios. This dimensionality also determines the internal consistency of the constraint set and the redundancy of a particular constraint in the context of pre-existing ones. While this procedure forms the underlying structure for a Geometric Constraint Management system, it has two major drawbacks. First, the computational effort could be potentially very large [1]. Second, the physical meanings of the variables used are not intuitive. As a consequence, the degrees of freedom of an entity are difficult to correlate to the variables in the Grobner Basis.

The limitations mentioned above arise because the special structure of the set of constraints of the GCS/SF problem was not exploited. Therefore, this chapter addresses these limitations by using a formulation devised to specifically express *spatial* constraints. It is possible to study them, and how they interact with each other, within the structure of the subgroups of the group of Euclidean displacements, $SE(3)$.

[1] Degrees of the polynomials in the Grobner Basis can grow at a double exponential rate in the maximum degree of the polynomials in the constraint set F

The subgroups of $SE(3)$ have been used by Herve [19] to characterize lower pairs (or joints) in mechanisms. Extending Herve's work, Angeles in [2, 3] has used operations between groups (intersection and direct product) to attempt the reduction of a mechanism to its essential degrees of freedom. Torras et al [34, 35] have studied mating constraints between objects in an assembly by using the formalization proposed by Herve.

In this chapter, we undertake the integration of the algebraic geometry-based approach, developed in prior chapters, with the formalisms provided by a group-theoretic analysis of the constraint set. This allows the two approaches to complement each other thereby reducing the effects of their individual disadvantages. The use of a group theoretic formulation introduces structure into an otherwise unstructured set of equations. By doing so, it has the potential of making the construction of the Grobner Basis more efficient. Further, the variables obtained by group theoretic formulation have direct physical meaning, producing a Grobner Basis that reflects the degrees of freedom of the entities. Viewed the other way, the Grobner Basis construction replaces the reduction based on group intersection. As mentioned in previous chapters, this reduction process deals only with the *topological* aspects of *trivial* constraints. The calculation of a Grobner Basis, though computationally expensive, is not limited to the trivial constraints and simultaneously enforces topological and geometrical consistency.

The next section discusses the group $SE(3)$ of the Euclidean displacements in E^3 and its subgroups. The conjugation classes developed by Herve [19] are then presented, and their relation to the constraints used in the formulation of the GCS/SF problem is developed. Next, a methodology for deriving constraint equations using the group theoretic analysis is introduced and illustrated with examples.

4.1 Subgroups of the $SE(3)$ Group and Canonical Variables

A **group** is a set S with a binary operation $\circ$ defined on S, which has the following properties [25, 26]:

- $G_1, G_2 \in S \rightarrow G_1 \circ G_2 \in S$ (closure property).
- $\exists I \in S, \quad s.t. \quad \forall G \in S, \quad G \circ I = I \circ G = G$ (identity element)
- $\forall G \in S \quad \exists H \in S, \quad s.t. \quad G \circ H = H \circ G = I$ (invertibility)
- $\forall G_1, G_2, G_3 \in S, \quad (G_1 \circ G_2) \circ G_3 = G_1 \circ (G_2 \circ G_3)$ (associativity)

$SE(3)$ is the group of Euclidean displacements in E^3. G_1, G_2 and G_3 represent displacements in $SE(3)$ and $\circ$ represents the composition of displacements. The particular meaning of the properties in the case of the group $SE(3)$ is recalled here:

- Two displacements G_1, G_2 applied in sequence produce a new displacement $G_3 = G_1 \circ G_2$.
- I is the null displacement in $SE(3)$. $G \circ I = I \circ G = G$
- For each $G \in SE(3)$ there is an inverse one G^{-1} which restores the affected entity to the original position $G \circ G^{-1} = G^{-1} \circ G = I$
- The effect of displacements is accumulative. If G's are applied in the order G_1, G_2 and G_3, the following sequences are identical (associativity):
  $(G_1 \circ G_2) \circ G_3 = G_1 \circ (G_2 \circ G_3)$

$SE(3)$ presents subsets which are groups themselves and which express certain common classes of displacements. They are called *subgroups*. For example, the subgroup of the rotations about a *given* axis u in the space, Ru, is a subset of $SE(3)$, and a group itself. Therefore, the composition of two sequential rotations about *the same axis* u is a rotation about u again (closure of groups). Although the classification of displacements into subgroups is theoretically sound, it is not useful because there are still infinitely many different subgroups of rotations, translations, etc. A contribution of Herve [19], was to *lump all the rotations, all the translations, etc* into sets, more populated than subgroups, called *conjugation classes*.

Table 4.1 Conjugation Classes and Their Canonical Forms

Dof	*Symbol*	*Conjugation Class*	*Canonical Subgroup*
1	Ru	Rotations about axis u	$\{twix(\theta)\}$
1	Tu	Translations along axis u	$\{trans(x,0,0)\}$
1	Hu,p	Screw movement along axis u, with pitch p	$\{trans(x,0,0)twix(px)\}$
2	Cu	Cylindrical movements along axis u	$\{trans(x,0,0)twix(\theta)\}$
2	Tp	Planar translations parallel to plane P	$\{trans(0,y,z)\}$
3	Gp	planar slidings along plane P	$\{trans(0,y,z)twix(\theta)\}$
3	So	Spherical rotations about center "o"	$\{twix(\psi)XTOYtwix(\phi)XTOYtwix(\theta)\}$
3	T	3D translations	$\{trans(x,y,z)\}$
3	Yv,p	Translating Screws axis v, pitch p	$\{trans(x,y,z)twix(px)\}$
4	Xv	3D translations followed by rotation about v	$\{trans(x,y,z)twix(\theta)\}$

Table 4.1, column 3, presents the classification that groups the displacements in $SE(3)$ into 10 conjugation classes. In order to identify the common structure in each class, Herve expressed all displacements in a standard way:

$$T^{-1}.S(x,y,z,\theta,\phi).T \tag{4.1}$$

with the term $S(x,y,z,\theta,\phi)$ being characteristic for each class. It appears in Table 4.1, column 4, and it is called *canonical* because it expresses the degrees of freedom of each class with a minimum of variables. [2] For example, using the canonical form, a rotation $R_w(\theta)$ about an axis w in the space can be written as:

$$R_w(\theta) = T(w)^{-1}.twix(\theta).T(w) \tag{4.2}$$

[2] In this Table, $twix(\theta)$ means a rotation about the X axis by θ; $XTOY$ means a rotation by 90^0 about the Z axis; $trans(x,y,z)$ indicates a general spatial translation.

where the $twix(\theta)$ part conveys the *topological* information about the class and its degrees of freedom, and $T(w) \in SE(3)$ stores the *geometrical* information about the actual position of the axis of rotation w.

The definitions presented above are intended to define an *equivalence* between displacements. They allow to state the equivalence between, for example, *rotations* (regardless of the axis), or likewise, *planar translations* (regardless of the plane). This equivalence is formalized next:

Definition: Given A, B, **subgroups** of the Euclidean **group** $SE(3)$, A is **conjugate** of B ($A \sim B$) iff $\exists T \in SE(3)$ such that $A = T^{-1}BT$. The relation $A \sim B$ is an equivalence relation. It is symmetric, reflexive and transitive. It defines equivalence classes called *conjugation classes.*

The T element above represents a rigid displacement. Therefore, two displacements A and B are equivalent iff a change of basis T converts one into the other. In this way, two displacements by 30 degrees are equivalent because all that differentiates them is a rigid transformation between their axes.

Equivalence classes have the property that the whole class can be represented by one element since all elements are equivalent via $\sim$. The representative, or *canonical* element, in this case results from making $T = I$ above. Therefore, the canonical element has the simplest possible *geometrical* component (I). Also, its *topological* part is minimal in the number of variables since they strictly reflect the physical degrees of freedom of the class. A list of the conjugation classes for the subgroups of $SE(3)$, their canonical representation [19] and their degrees of freedom are shown in Table 4.1.

4.1.1 Topological Manipulation of Trivial Constraints

This section briefly exposes the application of the concepts just introduced towards constraint manipulation.

4.1.1.1 Constraint Composition

If an entity B is forced to adopt positions which sacrifice degrees of freedom with respect to other entity A, it is said to be *constrained*. For example, if they can only rotate with respect to each other (1 dof), it is said that they are constrained by a rotational *joint*, or they hold a rotational constraint $G_1 = R_u(\theta)$. If another entity C is constrained with respect to B by G_2 the equivalent constraint between A and C has the structure $G_1 \circ G_2$. G_1 and G_2 are subgroups, extracted from the conjugation classes of Table 4.1. In general, $G_1 \circ G_2$ will not fall into any one of the conjugation classes. It will only do so in special cases in which the *geometry* of G_1 and G_2 allows the *reduction* of the constraint chain. For example, in

$$G_1 \circ G_2 = R_u(\theta_1) \circ R_u(\theta_2) = R_u(\theta_1 + \theta_2) \tag{4.3}$$

both rotations happen to have equal rotation axes (a geometrical circumstance). They can be reduced, and the reduction falls into the conjugation class of the "rotations". Sequences of constraints (which are also constraints) are expressed as:

$$G_1 \circ G_2 \circ G_3 .... \circ G_n \tag{4.4}$$

Trivial constraints are or can be reduced to *one* subgroup of $SE(3)$. The composition $G_1 \circ G_2$ is called the *direct product* of G_1 and G_2. If visualized as a kinematic situation, it can be thought of as a *serial* arrangement of joints.

4.1.1.2 Constraint Intersection

In the example above, if entities A and B are also constrained by G_3, the mobility of A with respect to B (or vice versa) becomes more restricted. Any possible movement must satisfy both G_1 and G_3. Therefore, this scheme represents an intersection of subgroups of $SE(3)$, which always forms a subgroup of $SE(3)$. Again, the resulting subgroup is

determined by the geometry of the constraints involved. For example,

$$T_u \cap T = trans(x, 0, 0) \cap trans(x_2, y_2, z_2) = T_u \tag{4.5}$$

If no mobility is left, the resulting subgroup is $\{I\}$ (a static structure). From this discussion it follows that an intersection of constraints can be thought of as a *parallel* arrangement of joints.

4.1.1.3 Table Look-up Algorithms for Constraint Reduction

In [19], tables are provided in which the result of composition and intersection of trivial constraints are tabulated. Two of such examples are shown in Table 4.2. For example, $Cu_0 \circ Cu_1$, illustrates the fact that the composition of trivial constraints in general produces a non-trivial one. This fact places a fundamental limitation in the reduction of constraint compositions and intersections by rewriting procedures based on tables such as Table 4.2. Thomas & Torras in [36] proposed a mechanical constraint network reduction that proceeds as follows: given a number of entities in the world and trivial constraints among them, a re-writing of the composition/intersection of constraints is based on pre-calculated results (Table 4.2). At each step, two adjacent constraints are considered. If a reduction is possible, the participant constraints are replaced by their equivalent. This process continues until all possible pairs are considered and no further reductions are possible. The final goal, which is not always attainable, is to reduce the whole constraint *graph* to a single, trivial constraint relating two entities. This approach is limited because it is unable to deal with geometric inconsistencies. For example, the intersection of two cylindrical groups Cu_0 and Cu_1, with $u_0 \parallel u_1$, produces a translational Tu_0 degree of freedom(Table 4.2). The physical situation corresponds to a rigidly linked pair of parallel pegs, entering into a pair of holes with parallel axes. The method would correctly establish that a translational joint is left (topological result). However, the *distance* between the axes has to be checked separately (geometric condition).

Table 4.2 Composition and Intersection of Trivial Constraints

Groups	*Conditions on Geometry*	*Intersection*	*Composition*
Cu_0, Cu_1	$u_0 \parallel u_1$	Tu_0	$Cu_0 \circ Ru_1$
$Tp0, Tp1$		$Tv0$ $v0 = P0 \cap P1$	T

Table 4.3 Entity Relations in the Form of Kinematic Joints

Macro	*Joint Chain*	*Kinematic Joints in Chain*	*Dof*
P-ON-P	S	spherical	3
P-ON-LN	$T_1 \circ S$	linear translation, spherical	4
P-ON-PLN	$T_2 \circ S$	planar translation, spherical	5
LN-ON-LN	C	cylindrical	2
LN-ON-PLN	$T_2 \circ Rv \circ Rw$	planar translation, revolute	4
PLN-ON-PLN	$T_2 \circ Rv$	planar translation, revolute	3

The important contributions of Herve, Angeles, and later, Thomas & Torras reside in (i) the representation of constraints as subgroups of the $SE(3)$ group of displacements; (ii) the formal definition of intuitive concepts such as *rotations*, *translations*, etc. in terms of conjugation classes; and (iii) the introduction of reduction techniques for the constraint graph that represents the GCS/SF problem. On the other hand, the limitations of the reduction techniques are the inability to deal with non-trivial constraints, and the failure in addressing the geometry of the problem.

This work uses the canonical form of conjugation classes developed by Herve to model the constraints of the GCS/SF problem. Therefore, they will be re-specified as shown in Table 4.3. The constraint reduction procedures developed by Thomas & Torras will be replaced by the more powerful algebraic geometric technique of Grobner Basis. The composition of the subgroups represented by the conjugation classes produces a series of matrix or polynomial equations, presented in the following sections.

4.2 Methodology with Canonical Variables

A **constraint** between two entities by definition maintains invariant certain relations between the constrained entities. For example, a *planar sliding, Gp*, allows two translational and one rotational degree of freedom, while still ensuring planar contact between the two parts. A *rotational, Ru* constraint preserves axial and radial relative distances, allowing one angular degree of freedom between the constrained entities.

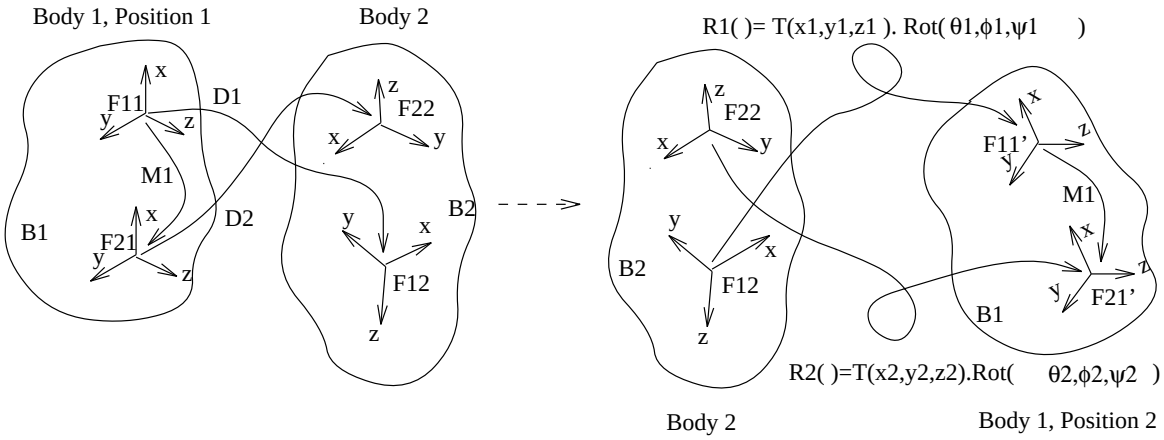

Figure 4.1 Two Body Example of Canonical Variable Modeling of the GCS/SF Problem

The GCS/SF problem is stated as a series of constraints R_i relating F_{i1} with F_{i2} as shown in Figure 4.1, (corresponding to a two body system). The $R_i()$ constraints are in general composed by translations $T()$ and rotations $Rot()$, as dictated by Tables 4.1 and 4.3. Body $B1$ contains two features, whose frames are F_{11} and F_{21}. Corresponding features in body $B2$ are F_{12} and F_{22}. The goal is to find a final position of $B1$ (assuming $B2$ is stationary), such that F_{11} relates to F_{12} and F_{21} relates to F_{22} satisfying the invariance dictated by $R_1()$ and $R_2()$ respectively. In the initial configuration, the relative position of entities F_{i1} and F_{i2} is a (known) displacement D_i. F'_{11} and F'_{21} denote frames F_{11} and F_{21} in the *final* configuration. The constraints $Ri()$ contain degrees of freedom to instantiate, satisfying the required relations while enforcing the rigidity of the two bodies. The modeling procedure follows:

(1) Transform F_{i1} by D_i, making it coincident with frame F_{i2}:

$$F_{11}.D_1 = F_{12}; \quad F_{21}.D_2 = F_{22} \tag{4.6}$$

(2) The above transformation in general destroys the rigid relation M_1 which is needed between frames F_{11} and F_{21}. The application of the desired constraints recovers the rigidity condition: $R_i()$:

$$F'_{11} = F_{11}.D_1.R_1(x_1, y_1, z_1, \theta_1, \phi_1, \psi_1); \quad F'_{21} = F_{21}.D_2.R_2(x_2, y_2, z_2, \theta_2, \phi_2, \psi_2) \tag{4.7}$$

(3) The rigidity conditions are enforced by recognizing that F_{11} and F_{21} must have the same relation in the initial and final configurations:

$$F_{11}.M_1 = F_{21}; \quad F'_{11}.M_1 = F'_{21} \tag{4.8}$$

(4) The above equations lead to a matrix equation which characterizes the cycle formed by transformations $D_1, D_2, M_1, R_1(), R_2()$:

$$F_{11}.D1.R_1().M_1 = F_{11}.M_1.D_2.R_2() \rightarrow D1.R_1().M_1 = M_1.D_2.R_2() \tag{4.9}$$

The above procedure can be generalized to the case in which there are several relations (constraints) $R_i()$ specified among bodies. The general situation is sketched in Figure 4.2. Once the constraint equations are obtained by this procedure, the construction of the Grobner Basis and its interpretation are carried out in the manner described by the constraint management algorithm discussed in the last chapter. The application of the above concepts is illustrated by the examples in next section.

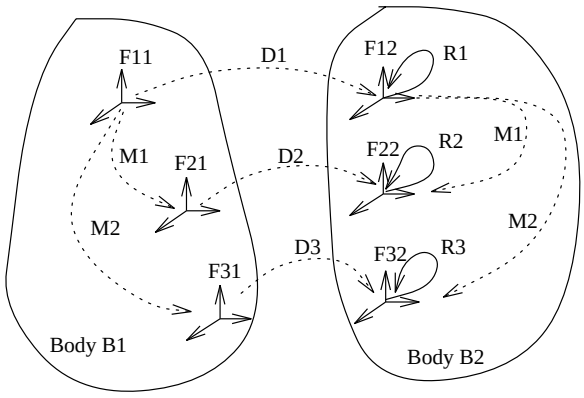

Figure 4.2 Methodology for Statement of Canonical form of Scene Feasibility Problem

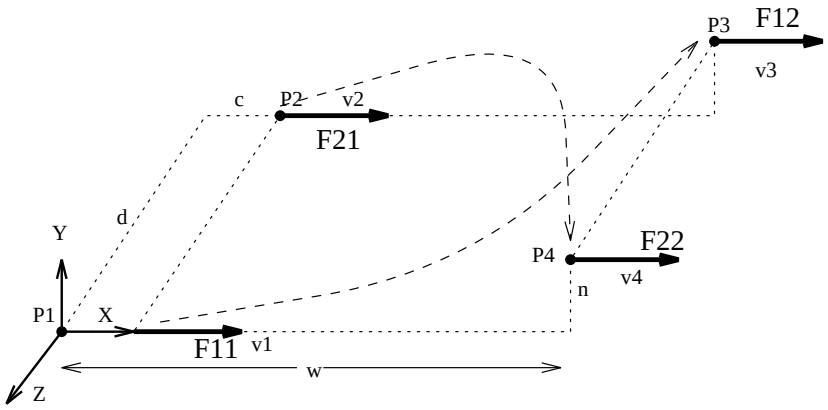

Figure 4.3 Simultaneous Line-to-Line Restriction between Pairs of Lines

4.3 Examples

This section presents examples of the modeling of the GCS/SF problem by using canonical and non-canonical formulations. It continues the discussion of the problem introduced in chapter 3, which involves the simultaneous enforcement of two $LN-ON-LN$ constraints (repeated in Figure 4.3). The use of canonical variables as an alternative for modeling the problem is presented. Another GCS/SF problem of increasing complexity is used to demonstrate the computational resources spent on the formulations.

4.3.1 Example 1. Solution with Canonical Variables

This example illustrates the methodology related to group theoretic analysis from the last sections. In this example, both constraints proposed are of the *trivial* type, as shown in Figure 4.3, in which the simultaneous enforcement of two $LN - ON - LN$ constraints appears.

The methodology mentioned above follows these steps:

(1) Frame LN_1 is placed onto frame LN_3: $LN_1.D_1 = LN_3$.
Frame LN_2 is placed onto frame LN_4: $LN_2.D_2 = LN_4$.
Where D_1, D_2 are the relative positions of LN_3 with respect to LN_1 and LN_4 with respect to LN_2 respectively. These positions (LN_3, LN_4) are not the final positions LN_1', LN_2'. The whole displacement of LN_1, LN_2 has to respect the constraints and the rigidity condition.

(2) The constraints are enforced by letting frame $LN_1.D_1$ undergo a cylindrical movement: $LN_1' = LN_1.D_1.Cu_1(\theta_1, x_1)$. Similarly, $LN_2' = LN_2.D_2.Cu_2(\theta_2, x_2)$.

(3) Record the rigidity condition between LN_1 and LN_2 in the initial configuration: $LN_1.M = LN_2$; M is the rigid link between LN_1 and LN_2.

(4) Enforce the rigidity condition M for the final configuration: $LN_1'.M = LN_2'$

Based on these equations, the matrix equations which govern this problem are (by eliminating LN_1, LN_2, LN_1', LN_2'):

$$D_1.Cu_1(\theta_1, x_1).M = M.D_2.Cu_2(\theta_2, x_2) \tag{4.10}$$

This matrix equation can be expanded in the form:

$$D_1.\begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & c_1 & -s_1 & 0 \\ 0 & s_1 & c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.M = M.D_2.\begin{bmatrix} 1 & 0 & 0 & x_2 \\ 0 & c_2 & -s_2 & 0 \\ 0 & s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

Where $c_1 = cos(\theta 1); s_1 = sin(\theta 1)$. Also, two equations of the form $c_1^2 + s_1^2 - 1 = 0$ are included.

Equation 4.10 represents a set of 12 polynomials and 6 variables. This equation set has the following (lexicographic) Grobner Basis:

$$\begin{gathered} \underline{s_2} - 1 = 0 \\ \underline{c_2} = 0 \\ \underline{s_1} + 1 = 0 \\ \underline{c_1} = 0 \\ \underline{x_1} - x_2 + c = 0 \end{gathered} \tag{4.12}$$

which is based on the order: $x_1 \succ x_2 \succ c_1 \succ s_1 \succ c_2 \succ s_2$. In these equations c is a constant. One can easily conclude that the zero-dimensionality property is violated by the absence of a polynomial whose $head()$ term (shown underlined in the Grobner Basis) contains a pure power of x_2. Indeed, x_2 is the variable which represents the remaining degree of freedom -the translational movement of the ensemble.

Notice that the variables in this set directly represent the degrees of freedom of entities, unlike the case of non-canonical variables. Further, they are fewer, therefore suggesting savings in the computing effort spent on the construction of the Grobner Basis. Table 4.4 presents some statistics corresponding to the examples shown.

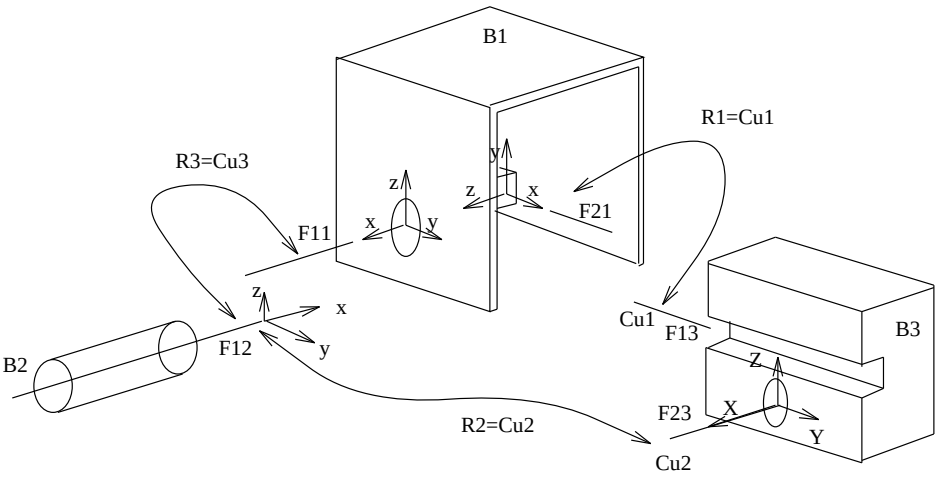

Figure 4.4 Three Body Assembly Producing Non-trivial Constraints

4.3.2 Example 2. Canonical vs. Non-canonical Variables. Non-trivial Constraints

This example demonstrates the advantages of using a canonical formulation of the problem, in cases containing a larger set of bodies, and (non-trivial) constraints. Figure 4.4 (adapted from [35]) shows the scenario being modeled. The following constraints are imposed on the entities:

- Line $F21 = (p_{21}, v_{21})$ is placed onto line $F13 = (p_{13}, v_{13})$ (Constraint R1)
- Line $F12 = (p_{12}, v_{12})$ is placed onto line $F11 = (p_{11}, v_{11})$ (Constraint R3)
- Line $F12 = (p_{12}, v_{12})$ is placed onto line $F23 = (p_{23}, v_{23})$ (Constraint R2)

Assuming that the body $B1$ is in the origin of the World, bodies $B2$ and $B3$ are in (unknown) positions $D2$ and $D3$ respectively.

4.3.2.1 Grobner Basis with Non-canonical Variables

The constraints mentioned above result in the following equations and conditions:

- $v_{21} \times v_{13} = 0; v_{21} \times (p_{21} - p_{13}) = 0$

- $v_{12} \times v_{11} = 0; v_{12} \times (p_{12} - p_{11}) = 0$
- $v_{12} \times v_{23} = 0; v_{12} \times (p_{12} - p_{23}) = 0$
- matrices $D2$ and $D3$ are orthonormal.
- $det(D2) = +1$, $det(D3) = +1$

This system results in the following solutions based on the Grobner Basis ($D2 = \{D2_{i,j}\}$ and $D3 = \{D3_{i,j}\}$):

$$\underline{D3_{34}} = 0 \tag{4.13}$$

$$\underline{D3_{33}} - 1 = 0$$

$$\underline{D3_{32}} = 0$$

$$\underline{D3_{31}} = 0$$

$$\underline{D3_{24}^2} - 10\, D3_{24} = 0$$

$$\underline{D3_{23}} = 0$$

$$\underline{5\, D3_{22}} - 5 + D3_{24} = 0$$

$$\underline{D3_{21}} = 0$$

$$\underline{5\, D3_{14}} + D3_{24} = 0$$

$$\underline{D3_{13}} = 0$$

$$\underline{D3_{12}} = 0$$

$$\underline{5\, D3_{11}} - 5 + D3_{24} = 0$$

$$\underline{D2_{34}} - 2 = 0$$

$$\underline{D2_{32}{}^2} + D2_{33}{}^2 - 1 = 0$$

$$\underline{D2_{31}} = 0$$

$$\underline{D2_{24}} - 5 = 0$$

$$\underline{D2_{23}{}^2} + D2_{33}{}^2 - 1 = 0$$

$$\underline{-D2_{22}} - D2_{23}\,D2_{32}\,D2_{33} + D2_{22}\,{D2_{33}}^2 = 0$$

$$\underline{D2_{22}\,D2_{32}} + D2_{23}\,D2_{33} = 0$$

$$\underline{D2_{22}\,D2_{23}} + D2_{32}\,D2_{33} = 0$$

$$\underline{{D2_{22}}^2} - {D2_{33}}^2 = 0$$

$$\underline{D2_{21}} = 0$$

$$\underline{D2_{13}} = 0$$

$$\underline{D2_{12}} = 0$$

$$\underline{D2_{11}} + D2_{23}\,D2_{32} - D2_{22}\,D2_{33} = 0$$

which is calculated based in the ordering: $D2_{11} \succ D2_{12} \succ D2_{13} \succ D2_{14} \succ D2_{21} \succ D2_{22} \succ D2_{23} \succ D2_{24} \succ D2_{31} \succ D2_{32} \succ D2_{33} \succ D2_{34} \succ D3_{11} \succ D3_{12} \succ D3_{13} \succ D3_{14} \succ D3_{21} \succ D3_{22} \succ D3_{23} \succ D3_{24} \succ D3_{31} \succ D3_{32} \succ D3_{33} \succ D3_{34}$ producing the following solution:

$$\begin{array}{lll}
D2_{11} \rightarrow -1 & D2_{12} \rightarrow 0 & D2_{13} \rightarrow 0 \\
D2_{21} \rightarrow 0 & D2_{24} \rightarrow 5 & D2_{31} \rightarrow 0 \\
D2_{34} \rightarrow 2 & D3_{11} \rightarrow -1 & D3_{12} \rightarrow 0 \\
D3_{13} \rightarrow 0 & D3_{14} \rightarrow -2 & D3_{21} \rightarrow 0 \\
D3_{22} \rightarrow -1 & D3_{23} \rightarrow 0 & D3_{31} \rightarrow 0 \\
D3_{32} \rightarrow 0 & D3_{33} \rightarrow 1 & D3_{34} \rightarrow 0 \\
D3_{24} \rightarrow 10 & D2_{22} \rightarrow -D2_{33} & D2_{23} \rightarrow -\sqrt{1 - {D2_{33}}^2} \\
D2_{32} \rightarrow -\sqrt{1 - {D2_{33}}^2} & &
\end{array} \tag{4.14}$$

The Grobner Basis (shown in Equation 4.13) is presented in triangular form, and the individual polynomials themselves have been arranged to have the $head()$ term (under the order presented above and underlined in the equations) in the leftmost position. The

examination of the Grobner Basis detects that variables $D2_{14}$ and $D2_{33}$ are missing in the $head()$ terms of polynomials. These two variables have a very definite role in the $D2$ matrix (as easily seen in this simple example): $D2_{14}$ represents a translational degree of freedom, while $D2_{33}$ represents a rotational degree of freedom about an unknown axis in space. This axis is determined by the eigenvalues and eigenvectors of the submatrix $Rot_2 = D2_{ij}(i = 1..3, j = 1..3)$ [5]. The solution implies (as expected) that body B3 is fixed while body B2 still has degrees of freedom left, represented in the variables $D2_{14}, D2_{33}$. Notice that in this case, non-instantiation of $D2_{33}$ immediately spreads to $D2_{32}, D2_{23}, D2_{22}$, since these values control the eigenvalues and eigenvectors of the matrix Rot_2. However this information is not self-evident from the solution set.

4.3.2.2 Grobner Basis with Canonical Variables

In this case the system of matrix equations can be stated as:

$$F_{21}.C_{u1}(x_1, \theta_1).F_{13}^{-1} = F_{11}.C_{u3}(x_3, \theta_3).C_{u2}(x_2, \theta_2).F_{23}^{-1} \tag{4.15}$$

Using an ordering $x_1 \succ x_2 \succ x_3 \succ s_1 \succ c_1 \succ s_2 \succ c_2 \succ s_3 \succ c_3$ produces a (lexicographic) Grobner Basis:

$$\begin{aligned} \underline{{s_3}^2} - 1 + {c_3}^2 &= 0 \\ \underline{c_2} - c_3 &= 0 \\ \underline{s_2} + s_3 &= 0 \\ \underline{c_1} - 1 &= 0 \\ \underline{s_1} &= 0 \\ \underline{x_2} + x_3 &= 0 \\ \underline{x_1} &= 0 \end{aligned} \tag{4.16}$$

The use of canonical variables immediately gives information on the degrees of freedom: because of its role in the group equations ($C_{u3}(x_3, \theta_3)$ represents $Cos(\theta_3)$, and the

Table 4.4 Statistics for Examples. Non-canonical vs Canonical Variables

Example	*Variable Type*	*Variables*	*Equations*	*GB Size*	*Time (secs)*
Example 1	Non-canonical	12	20	16	1.53
Example 1	Canonical	6	14	6	0.25
Example 2	Non-canonical	20	30	24	6.08
Example 2	Canonical	9	15	7	0.51

Grobner Basis indicates that it is dependent on c_2. Meanwhile, x_1, c_1, s_1 are completely instantiated, showing that the position of the body $B3$ is fixed. Body $B2$ is free to rotate about and translate along axis $F12$. This is confirmed by the fact that x_2 and s_2 do not appear in the $head()$ terms in polynomials of the Grobner Basis, and this fact indicates that they are the variables representing the remaining degrees of freedom. In this example again it is seen that the canonical variables present a convenient way to simplify the equations and give geometric meaning to the polynomial solution process.

Table 4.4 presents statistics for the different examples developed. It shows the number of variables involved in the modeling, the size of the equation set which expresses the constraints and the size of the corresponding Grobner bases. Finally, it shows the execution times for the canonical and non-canonical modeling for each example.

4.4 Summary

In previous chapters, a systematic method for managing geometric constraints was presented. First, the application of the Grobner Basis in characterizing the solution space of a set of constraints was discussed. One potential drawback of a direct application of this algebraic geometric technique is the growth of computational effort with problem size. This problem was addressed in this chapter by the choice of a convenient set of variables (canonical) dictated by the conjugation classes of the subgroups of the group $SE(3)$ of the Euclidean displacements.

Canonical variables present a compact representation of the GCS/SF problem constraints and have a direct physical meaning. Therefore, they facilitate the interpretation

of variables in terms of the degrees of freedom, allowing an easier analysis of the solution space. The statistics on computational effort presented suggest that canonical formulation presents advantages over its non-canonical counterpart. Non-canonical variables, however, cannot be entirely dismissed since they may present advantages in situations where a small number of bodies have many interactions between themselves. Further investigation will be presented later, which characterizes the systems of constraints that are efficiently modeled by each method. The application of Grobner Basis to the GCS/SF allows the consideration of geometrical as well as topological aspects in the constraint set. It is not limited to trivial constraints as is the case with techniques associated with group theoretic approaches. The next chapters deal with the possibility of pre-processing the local parts of the constraint network, by using the Grobner basis method itself, which will produce a reduced set of constraints.

CHAPTER 5

Constraint Reduction

5.1 Introduction

In previous chapters, the methodology for the formulation of the GCS/SF problem in terms of non-canonical variables was discussed. Being a first attempt at solving the problem, this formulation resulted in a large number of variables. Also, it presented difficulties for determining of the mapping between variables and the physical degrees of freedom of the entities. As a response to these shortcomings a second formulation was proposed, based on the subgroups of the Special Euclidean Group of displacements in E^3, $SE(3)$. By exploiting the fact that the GCS/SF problem originates from a geometric domain, this formulation uses variables that directly relate to the degrees of freedom of the entities. As a consequence, the size of the GCS/SF problem was reduced and the remaining degrees of freedom could be determined in a straightforward manner. In spite of this improvement, the computational complexity of the GCS/SF problem requires additional efforts to exploit the physical characteristics of the set of constraints.

This chapter discusses how the structure of a particular instance of a GCS/SF problem might be exploited to obtain a further reduction of the computational effort of producing its Grobner Basis. The general idea is to identify local subproblems whose solution might be easily obtained and then merged to produce a solution to the overall set of equations of the GCS/SF problem. These techniques, called *Divide & Conquer* techniques in this investigation, will be discussed here.

It will be shown that the Spatial Constraint (SC) graph structures the set of equations of the GCS/SF problem. Additionally, the *basic cycles* in the SC graph map into local

subproblems which can be solved separately, thus contributing to the complete solution. Therefore, the issue of identification and classification of GCS/SF subproblems requires a theoretical background which is related to the topological [1] properties of graphs. This chapter addresses such theory and translates it into algorithms for identifying subgraphs in the SC graph. An example of the application of Divide & Conquer techniques is presented at the end of this chapter.

5.2 Background

The SC graph conveys the topological and geometrical information of the GCS/SF problem. In the context of the SC graph, cycles represent constraint intersections. They are named by the corresponding constraint sequences. For example, $C_5-C_1-C_7-C_3-C_6$ is composed by constraints C_5, C_1, C_7, C_3 and C_6.

It is assumed that the graphs in the discussion have the following characteristics:

(1) $G = (E, V)$ is an undirected graph, with node set $V = \{v_1...v_n\}$ and edge set $E = \{e_1....e_m\}$. Although the edges in the SC graph are *directed*, edges representing constraints can be inverted; the corresponding degrees of freedom θ, x, y, z change sign, and the order of the matrices in the chain is inverted:

$$(C_1(\theta).C_2(\phi))^{-1} = C_2(\phi)^{-1}.C_1(\theta)^{-1} = C_2(-\phi).C_1(-\theta) \tag{5.1}$$

Therefore the structural properties of the SC graph can be calculated based on the assumption of non-directedness.

(2) G has only one component.

(3) Every edge $e_i \in E$ belongs to a cycle. This assumption follows from the fact that an open chain of constraints does not present interest from the point of view of

[1] In the context of graph theory, *topology* refers to the connectivity of the graph. In the context of the *GCS/SF* problem it includes the *constraint network* between entities and the *type* of constraint relations between them.

solution space reduction, since it doesn't include the simultaneous satisfaction of constraint conditions.

In an abuse of notation, and since vertices in graphs are implied by the edges in which they participate, sometimes the graph is equated to its set of edges.

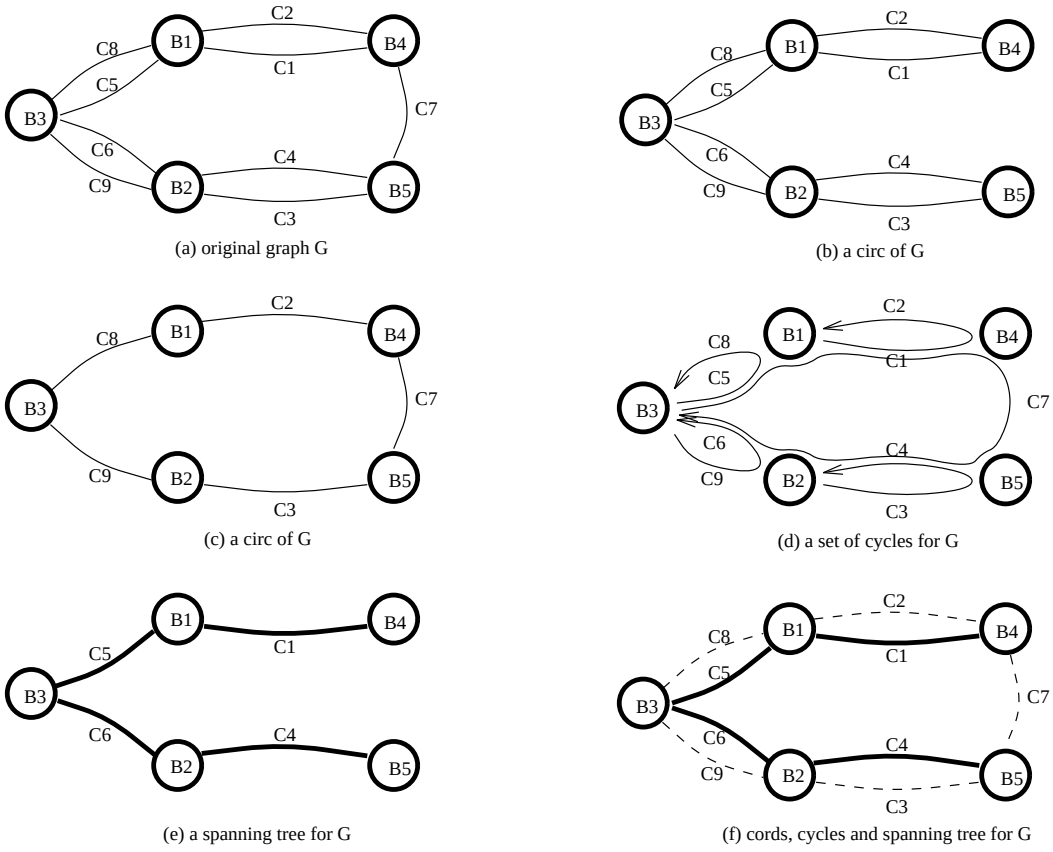

Figure 5.1 Sub-Graphs of the Spatial Constraint Graph

Under the assumption that the graph of Figure 5.1(a) represents a Spatial Constraint graph, the cycle equations can be stated as part of the solution of the GCS/SF problem. As an example, the simultaneous satisfaction (intersection) of constraints C_i indicates that

$$C_8 \circ C_2 \circ C_7 = C_9 \circ C_3 \tag{5.2}$$

In a similar manner, all the relevant equations should be expressed, and no redundant equations should be included. Several problems are immediately evident: (i) how to determine formally when an equation is redundant; (ii) how to determine formally that *all* the relevant equations have been included; (iii) which set is more convenient for solving the GCS/SF problem among the many sets of complete, non-redundant equations; and (iv) how to obtain it?

Since the equations of the GCS/SF problem directly relate to the cycles of the SC graph, the questions stated above can be expressed in other ways: (i) what does it mean for a cycle to be redundant and how to identify it; (ii) how to enumerate the cycles of a graph; (iii) which set of cycles in the SC graph has special meaning in the context of the GCS/SF problem; and (iv) what kind of cycle partitions exist in graphs and how expensive is to extract them. The theoretical background and algorithms presented in this chapter are essential in *properly stating* the GCS/SF problem to answer the questions above. Beyond this consideration, it will be seen that careful manipulation of the SC graph presents attractive ways to *improve* the efficiency of the solution of the GCS/SF problem.

5.3 Definitions

To develop a basis for the discussion ahead, some standard definitions([12, 11]) are reviewed here:

Spanning Tree: In a connected graph $G = (V, E)$, a spanning tree $ST(G)$ is a subgraph of G which contains all the nodes of V and a set of arcs which make it connected, and acyclic (Figure 5.1 (e)).

Branch b_i**:** Is an edge in $ST(G)$ (for example, c_1, c_5, c_6 and c_4 are branches in Figure 5.1 (e)).

Cord c_i**:** Is an edge in $G - ST(G)$ (for example, c_2, c_3, c_7, c_8 and c_9 are cords in Figure 5.1 (f)).

Cord set: Is the set of all edges removed from G to convert it into a spanning tree. Therefore, $CS = G - ST(G)$.

Cycle: A cycle (or loop) $L_i = \{e_w, e_{w+1}, .., e_k\}$ is a subgraph of G whose nodes have exactly two incident edges.

Set of Basic Cycles: A set of cycles $SBC = \{L_1, L_2, L_3, .., L_m\}$ in a graph $G(V, E)$ is called a *Basic Set of Cycles* if:

(1) $\forall \Gamma$ cycle in G, $\Gamma = L_i \oplus L_k ... \oplus L_w$. It means Γ can be expressed as a *linear* combination of the cycles L_i in SBC.

(2) No L_k basic cycle in SBC can be written as a linear combination of other cycles in SBC.

For example, $SBC = \{\{c_1, c_2\}, \{c_3, c_4\}, \{c_5, c_8\}, \{c_6, c_9\}, \{c_1, c_5, c_6, c_4, c_7\}\}$ in Figure 5.1 (d).

Circ: A *circ* is an undirected graph $C = (V_c, E_c)$ whose nodes have even degrees; all of them receive an even (nonzero) number of incoming arcs (Figure 5.1 (b),(c)).

Ring Sum: $\oplus$ is the ring sum operation between edge subsets F and H of E ($F \subset E$, $H \subset E$):
$H \oplus F = (H \cup F) - (H \cap F)$.

5.3.1 Structure of the Set of Circs of a Graph

To present the algorithms for extraction of a basic set of cycles in a graph and to give them theoretical support, a short discussion follows concerning the mathematical structure of subgraphs in a graph, and in particular, the classes of circs and cycles. In this review, standard graph theorems and lemmas relevant to this research are presented as propositions, with no formal proof. For a deeper insight the reader can consult [11].

Proposition 1

The set S of all subgraphs of graph $G = (V, E)$ is an abelian group with operation $\oplus$.

Each element G_i of S is a subgraph whose inverse is G_i itself. The element identity is the empty graph Φ (see [33, 11]).

Definition: Consider the set $F = \{0, 1\}$ with operations $+_2$ (addition module 2) and $*_2$ (multiplication module 2). F has the properties of a field, and serves to introduce the scalar product $\ . \ : F \times S \longrightarrow S$ which can be defined as:

$$0.G_i = \Phi; \quad 1.G_i = G_i \tag{5.3}$$

Comment: Informally, the operation $0.G_i$ means "neglect subgraph G_i", while the operation $1.G_i$ means "choose subgraph G_i". This operation is necessary when a set of subgraphs $\{G_1, G_2 ... G_e\}$ is used to build any other subgraph G_m of G. The following proposition formalizes that concept.

Proposition 2: Let $(S, \oplus)$ be the abelian group, $(F, +_2, *_2)$ the field and . the *scalar* multiplication as defined above. Then, S is a linear space over F (see [33, 11]).

Comment: Given the graph $G = (V, E)$ with edge set $E = \{e_1, e_2, ..e_e\}$, a possible basis for the linear space S could be the *canonical set* (in the sense of linear algebra) $\{G_1 = \{e_1\}, G_2 = \{e_2\}, ...G_e = \{e_e\}\}$. In this way, any subgraph $G_w \in S$ can be expressed as:

$$G_w = (\alpha_1.G_1) \oplus (\alpha_2.G_2) \oplus ... \oplus (\alpha_e.G_e) \tag{5.4}$$

by choosing convenient values for the $\alpha_i \in F$. Therefore if a subgraph s_i contains the edges e_k, e_j, e_w, then s_i can be written as:

$$s_i = (0.G_1) \oplus ... \oplus (1.G_k) \oplus (1.G_j) \oplus (1.G_w) ... \oplus (0.G_e) \tag{5.5}$$

The following propositions move the emphasis from the general set of all subgraphs to the particular case of circs, from which the cycles are a subset. From Figure 5.1 (b) and (c) several formal properties can be observed: (i) a circ is either a cycle or the (set) union of edge-disjoint cycles; and (ii) the ring sum of two cycles is either a cycle or the union of edge-disjoint cycles. The application of these properties is the construction of the

set of circs. In order to parallel the construction procedure used for general subgraphs, it should be noticed that (iii) the set of all circs in a graph is an abelian group under $\oplus$. With these foundations, the construction of the set of the circs in a graph can be undertaken.

Proposition 3: The set of all circs in a graph is a subspace S_c of S (see [33, 11]).

Once the elements which allow the construction of the subspace of circs are identified, the dimension of such subspace becomes relevant. This dimension indicates how many independent cycles are required to obtain a complete coverage of the set of cycles in a graph. Therefore, it is a termination condition for a basis construction algorithm.

The extraction of a set of independent cycles is closely linked to the determination of a spanning tree T for a graph G. It is a well known fact [12, 33, 11] that a spanning tree has $|V|-1$ edges. Therefore, there are $|E|-(|V|-1)$ edges in the corresponding set of cords. Since the T has no cycles but covers all the nodes, the addition of each cord c_i produces exactly one cycle in $T \cup \{c_i\}$. Each cycle is independent from the others because it contains a new cord. Therefore, there are $|E|-(|V|-1)$ different, independent cycles. The following propositions formalize such a concept.

Proposition 4: Let $c_i \in G - ST(G)$ be a cord. Then, $ST(G) + \{c_i\}$ contains exactly one cycle $\{c_i, b_k, b_m ..., b_w, c_i\}$ (see [33, 11, 12]).

Proposition 5: Let G be a one component graph with $G = (V, E)$. Then, there are exactly $|E| - |V| + 1$ cords in $G - ST(G)$ (see [11, 12]).

Proposition 6: The set of all circs in a graph is a subspace of dimension $|E|-|V|+1$ (see [11, 12, 38]).

Proposition 7: Let $cycle\_set = \{L_1, L_2, L_3, .., L_{|E|-|V|+1}\}$ be a set of cycles, with the following characteristics: each cycle L_i is the form $L_i = \{c_j, b_k, b_m ..., b_w, c_j\}$ with c_i a cord and the b_j being branches of a spanning tree ST. Then, $cycle\_se$ contains $|E| - |V| + 1$ independent cycles, and $cycle\_set$ constitutes a basis for the set of cycles for the graph G (see [33, 11, 12, 38]).

Figure 5.1 illustrates these concepts; the spanning tree in (e) contains $|V| - 1 = 4$ edges. This spanning tree produces a cord set of $|E| - (|V| - 1) = 5$ elements, which

immediately implies that the S_c subspace has dimension 5. Given the spanning tree of (e), restoring each cord c_i (marked in dashed lines in (f)) produces an independent cycle. Notice that although the basis produces the *circ* subspace, since the cycles are a subset of the circs, having a basis for the circs ensures the production of the cycles.

The facts mentioned above suggest that the construction of a basic set of cycles for a graph can be achieved by obtaining a spanning tree T and the set of corresponding cords (sometimes called *cotree* T'). Each time a cord c_i is added to T, one and only one cycle is produced. Since exactly $|E| - |V| + 1$ cycles are needed and there exist $|E| - |V| + 1$ cords, it follows that the set of cycles obtained in this way serves as a basis for the set of circs (and therefore cycles) of the graph. Obviously, the equations for the GCS/SF problem only need to be written for the cycles which form the basis for the S_c subspace in the SC graph; any other set of equations can be written as a linear combination of the equations for the set of basic cycles.

5.4 Extraction of the Basic Set of Cycles

The following algorithm makes use of the fact that once $|E| - |V| + 1$ independent cycles have been identified they form a basis for the set of circs of the graph. As a by-product, the original graph G has been partitioned into a spanning tree T and the corresponding cotree T'.

For the present application, namely the solution of the GCS/SF problem, it is important that the cycles belonging to the basic set of cycles be as small as possible. This requirement allows the application of the Divide & Conquer method to sets of polynomials involving easily reducible solution spaces. These partial solutions can be used to attack the complete GCS/SF problem.

In this section, an algorithm is considered which constructs the basic set of cycles in a graph by using a low-depth spanning tree T. In a spanning tree T every cord completes a cycle that in the worst case has length $2H + 1$, where H is the depth of the tree. By

using a low-depth spanning tree, the largest cycle length is limited, therefore producing a set of small cycles.

5.4.1 Extracting the Basic Set of Cycles Given a Spanning Tree

If the existence of a spanning tree T for a graph G is assumed, the construction of the actual cycles would follow, with the procedure $cycles\_from\_tree(T : tree, T' : graph; C : set\ of\ graph)$ which takes the partition of a graph into a tree T and a cotree T' and constructs the set of cycles C. With no loss in generality it can be assumed that T has indeed a tree data structure which can be used to retrieve the ancestors of a node in the tree all the way to the root. This assumption facilitates the task of extracting the cycles from the original graph G, by using the following algorithm:

```
procedure cycles_from_tree(T : tree, T' : graph; C : set of graph)
 0 {
 1   C = {};
 2   do (T' ≠ {}) →
 3        c_i = first_cord(T');
 4        [v_1, v_2] = extremes_of_edge(c_i);
 5        l_1 = path_to_root(T, v_1);
 6        l_2 = path_to_root(T, v_2);
 7        [l_common, tail_1, tail_2] = common_path(l_1, l_2);
 8        cycle = [invert(tail_1), tail_2, {c_i}];
 9        C = C ∪ {cycle};
10        T' = T' − {c_i};
11   od
12 }
```

Intuitively, $cycles\_from\_tree(T, T', C)$ takes each cord c_i in the cotree T' (line 3), and determines the cycle that such a cord completes in the spanning tree T. For this purpose, the first node in the tree that is common ancestor of nodes v_1 and v_2 is searched by determining the path from root to v_1 (l_1) and from root to v_2 (l_2) (lines 5, 6). If these

paths are cross-examined, the common ancestors will be found (l_{common}), as well as the two non-common paths, $tail_1$ and $tail_2$ to v_1 and v_2 respectively (line 7). If the two tails are glued together with the closing edge c_i, the cycle is determined (line 8).

5.4.2 Extraction of a Low-Depth Spanning Tree

In order to guarantee the input for the procedure $cycles\_from\_tree()$, a pre-processing procedure $spanning\_tree()$ extracts a spanning tree T from a graph G.

```
procedure spanning_tree(G : graph; T : tree)
 0 {
 1   head = max_degree_node(G);
 2   T = {};
 3   visited = {};
 4   to_visit = {head};
 5   do (to_visit ≠ {}) →
 6        node = max_degree_node(to_visit);
 7        branching = incident_edges(node);
 8        do (branching ≠ {}) →
 9             edge = first_element(branching);
10             v = opposite_extreme(edge, node);
11             if ((v ∉ visited)and(v ∉ to_visit)) →
12                  to_visit = to_visit ∪ {v};
13                  T = T ∪ {edge};
14             fi
15             branching = branching − {edge};
16        od
17        visited = visited ∪ {node};
18        to_visit = to_visit − {node};
19   od
20 }
```

The algorithm is biased to heuristically extract a low-depth spanning tree. As a start, it chooses the root of the tree to be a large-degree node (line 3), and each time a node in a set is considered for branching, the chosen candidate is also the one which presents the

largest degree. In this heuristic way, by forcing a large branching in the tree, its depth would be expected to be low.

spanning_tree() constructs a list of the nodes that have to be visited, along with the ones already branched. The strategy of developing the tree lies in the administration of such a list (line 6). If the list is managed strictly as a queue, a breadth-search strategy would result. If instead a stack management is used, a depth-first strategy results. Departing from these two alternatives, in this research the list is sorted by size of the degree of the nodes involved.

Each time a node is included in the tree (line 6), its neighbor nodes are scheduled to be visited (line 12) if they haven't been visited already (line 11). The algorithm stops when the *to_visit* set is empty (line 5).

An execution of the *spanning_tree*() routine for the graph in Figure 5.1 would start with node B_3 (whose degree is 4) and continue with nodes B_1 and B_2, eventually producing the spanning tree $T = \{C_1, C_5, C_6, C_4\}$. The cotree $T = \{C_7, C_2, C_8, C_9, C_3\}$ provides the cords, each one of them completing a cycle in the graph G. In this case four of the completed cycles have length 2 and one has length 5.

5.4.3 Complexity Analysis

In the *spanning_tree()* procedure, the external loop (line 5) is executed $O(V)$ times. Internal to that loop, the largest operation is the branching loop (line 8) with complexity $O(E)$. Inside such a loop there is an $O(E)$ operation in $first\_element()$ and an $O(V)$ operation in the if (line 11). From these considerations it can be concluded that the complexity of the *spanning_tree*() algorithm is $O(VE^2 + EV^2)$. The analysis for the *cycles_from_tree()* procedure indicates that since T is assumed to have tree structure, the operation *path_to_root*() can be completed in $O(V)$ (lines 5, 6), and the execution of *common_path*() in $O(V^2)$ steps. Next, the two legs $tail_1, tail_2$ are appended, together with the cord c_i to make the cycle (line 8). Since the loop (line 2) has to be performed exactly $E - V + 1$ times, the complexity of this procedure is $O(EV^2)$ (assuming $E > V$).

From the considerations above, it can be concluded that the complexity of the extraction of a set of basic cycles from a graph is $G = (V, E)$ is $O(VE^2 + EV^2)$.

5.5 Constraint Reduction for the GCS/SF Problem

Before starting the application of the algorithm just presented, a short recapitulation may be useful. In Chapter 4 and Chapter 5, methodologies for the statement of the GCS/SF problem using non-canonical and canonical variables were discussed. Regardless of the methodology used, the complete and non-redundant set of constraints has to be used in the production of a set of polynomials to be input to the Grobner Basis algorithm. The partition of the original GCS/SF problem into a basic set of cycles for the SC graph produces exactly a complete and non-redundant coverage of the SC graph (and of the problem). Additionally, the cycles of the graph represent local GCS/SF subproblems. By analyzing these subproblems, conclusions about local scenes can be drawn, lowering the overall computational expenses. In some domains of application, such as assembly planning, sub-assemblies can be identified by this method. Therefore, the partition of the GCS/SF problem presents direct applications in CAD / CAM environments. The strategy called here *Divide & Conquer* comprises the subdivision of the GCS/SF problem, the analysis of its local sub-problems and the integration of such analysis into the solution for the original system. In this section such a strategy is developed, and a variant of it, called *Incremental Instancing*, is discussed. An example illustrates the application of these concepts.

5.5.1 Divide & Conquer Algorithm

The Divide & Conquer (**D&C**) algorithm showed below assumes the existence of a fundamental set of basis cycles for the SC graph. It extracts the polynomial equations for each cycle L_i (lines 2, 4) and calculates its Grobner Basis gb_i (line 5). The equations obtained in this way are put together into the set $full\_equations$ (line 7), whose Grobner

Basis is finally calculated. Obviously, if any one of the g_i sets shows any inconsistency ($gb_i = \{1\}$), the process should stop (line 9).

```
procedure Divide_and_Conquer(G set of Graph )
    0 {Pre: G = {L_1, L_2, ..L_k} basic cycles in Spatial Constraint Graph}
    1   full_equations = {};
    2   do not_empty(G)
    3   {Inv:    full_equations has same roots as {L_1, L_2, ..L_i} }
    4       L_i = next_cycle(G);
    5       gb_i = GB(equations(L_i), ≺_l) ;
    6       if (gb_i ≠ {1}) ⟶
    7           full_equations = full_equations ∪ gb_i;
    8       else ⟶
    9           exit;
    10      fi
    11      G = G − {L_i};
    12  od
    13  full_GB = GB(full_equations, ≺_l) ;
    14 {Post:full_GB is the Grobner Basis for equations(G) }
```

The rationale behind the partition technique just discussed is based on several facts: (i) the individual gb_i are Grobner Basis for the polynomials representing each basic cycle. Therefore, they have no internal redundancy; (ii) local inconsistencies are filtered *before* the full GCS/SF problem is addressed; (iii) local solutions to subproblems can be found and used towards the solution of the full problem; and (iv) the g_i sets represent an already (triangularly) ordered set of polynomials. Although it is not within the scope of this investigation to examine the details of Grobner Basis calculation, it is possible that in later work the pre-ordering in the individual Grobner Bases could be exploited to speed up the processing of the full set.

5.5.2 Incremental Instancing Algorithm

The *Incremental Instancing* (**II**) method is a variant of the D & C technique, in which variables that can be given a value by the characteristics of the local constraint scenario are instanced immediately, therefore progressively reducing the size of the variable and polynomial sets.

```
procedure Incremental_Instancing(G set of Graph )
 0 {
 1 {Pre: G = {L_1, L_2, ..L_k} basic cycles in Spatial Constraint Graph}
 2   full_equations = {};
 3   free_variables = {};
 4   instanced_variables = {};
 5   do not_empty(G)
 6   {Inv:    full_equations has same roots as {L_1, L_2, ..L_i} }
 7        L_i = next_cycle(G);
 8        V_i = variables(L_i) − instanced_variables;
 9        gb_i = GB(equations(L_i), V_i, ≺_l) ;
10        if (gb_i ≠ {1}) ⟶
11             instanced_variables = instanced_variables ∪ instanced_vars(V_i, gb_i);
12             full_equations = full_equations ∪ instanced_form(gb_i);
13        else ⟶
14             exit;
15        fi
16        G = G − {L_i};
17   od
18   free_variables = all_variables(G) − instanced_variables ;
19   full_GB = GB(full_equations, free_variables, ≺_l) ;
20 {Post:full_GB is the Grobner Basis for equations(G) }
21 }
```

In this algorithm a set called *instanced_variables* is maintained which contains the variables that have taken a value at any point in the execution. Subsequently, only variables not contained in this set can be considered for Grobner Basis calculation (lines 8, 9). If a Grobner Basis is successfully calculated for a cycle (line 10), the set of instanced variables is augmented by its contribution (line 11), and the general set of polynomials,

full_equations , is augmented by the *partially* instanced version of its set of polynomials gb_i (line 12). When the solution of the overall GCS/SF problem is finally undertaken, only the free variables and the instanced version of the individual Grobner Bases gb_i are used (lines 18, 19).

5.5.3 Application of Constraint Reduction. Cartesian Table

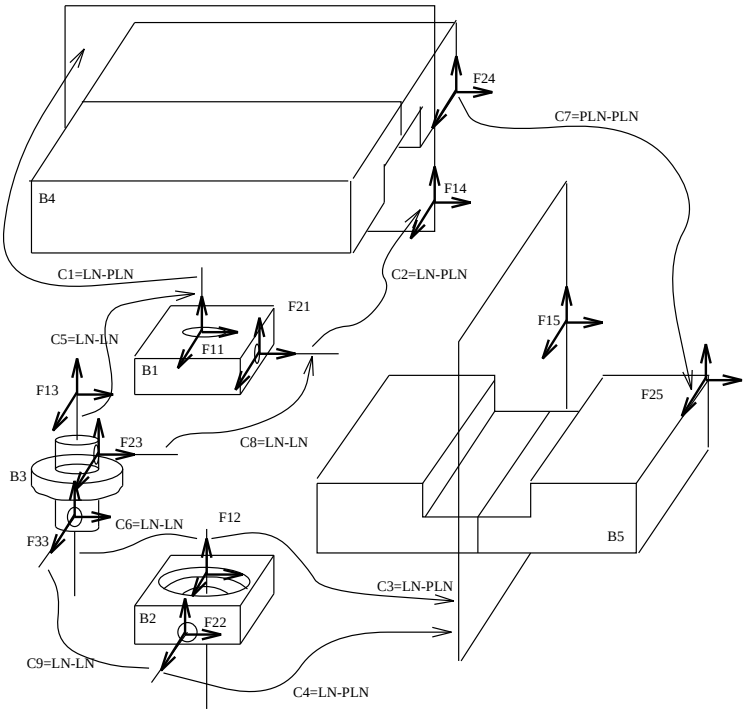

Figure 5.2 Piece Disassembly of Cartesian Table

The Cartesian Table example is presented to illustrate how the theory for partitioning and solving the GCS/SF problem discussed in this chapter can be applied. Given the constraint configuration (the topological information), and the dimensions of the entities (the geometrical part), the goal is to determine the degrees of freedom of the different bodies in the mechanism. The Cartesian Table (see Figure 5.2) is intended to undergo orthogonal movement, which involves two translational degrees of freedom, therefore producing a planar translation between bodies B_4 and B_5. Notice that with the specified

Table 5.1 Joint List of the Cartesian Table

Constraint	*Constraint Type*	*Elements*	*Canonical Representation*
C_1	$LN - PLN$	F_{11}, F_{14}	$R_u(\theta_1) \circ T_p(y_1, z_1) \circ R_u(\phi_1)$
C_2	$LN - PLN$	F_{21}, F_{14}	$R_u(\theta_2) \circ T_p(y_2, z_2) \circ R_u(\phi_2)$
C_3	$LN - PLN$	F_{12}, F_{15}	$R_u(\theta_3) \circ T_p(y_3, z_3) \circ R_u(\phi_3)$
C_4	$LN - PLN$	F_{22}, F_{15}	$R_u(\theta_4) \circ T_p(y_4, z_4) \circ R_u(\phi_4)$
C_5	$LN - LN$	F_{13}, F_{11}	$C_u(\theta_5, x_5)$
C_6	$LN - LN$	F_{13}, F_{12}	$C_u(\theta_6, x_6)$
C_7	$PLN - PLN$	F_{24}, F_{25}	$G_P(\theta_7, y_7, z_7)$
C_8	$LN - LN$	F_{23}, F_{21}	$C_u(\theta_8, x_8)$
C_9	$LN - LN$	F_{33}, F_{22}	$C_u(\theta_9, x_9)$

constraints, the bodies B_1, B_2 and B_3 have zero degrees of freedom relative to each other. This fact, together with constraints C_1, C_2, C_3 and C_4, forces the planes F_{15} and F_{14} to remain perpendicular to each other. An additional G_P (planar sliding) constraint forces planes F_{25} and F_{24} to stay in contact, therefore producing the desired $X - Y$ movement. The types of joints present in this mechanism appear in Table 5.1. The features F_{ij} involved with each constraint C_k appear in column 3. The compositions of subgroups of $SE(3)$ that constitute each C_k appear in column 4. Notice that this example includes non-trivial constraints such as C_1, C_2, C_3 and C_4.

5.5.3.1 A Partition of the Cartesian Table Problem

Figure 5.3 shows the SC graph for the Cartesian Table and a simple abstraction of it. The *Divide* part of the Divide & Conquer strategy includes the identification and solution of local GCS/SF subproblems. The set of basic cycles contains four cycles of length 2, and one cycle of length 5. The SC graph presents $|V| = 5$ nodes (entities) and $|E| = 9$ edges (constraints). Since the set of basic cycles must have $|E| - |V| + 1 = 5$ cycles, it follows that, since the cycles mentioned above are independent, they constitute a basis for the set of circs (and cycles) of the graph. In this example, the algorithm discussed for partition of the SC graph produced the following set of basic cycles:

$$SBC = \{\{C_1 - C_2\}, \{C_3 - C_4\}, \{C_6 - C_9\}, \{C_8 - C_5\}, \{C_5 - C_1 - C_7 - C_3 - C_6\}\} \quad (5.6)$$

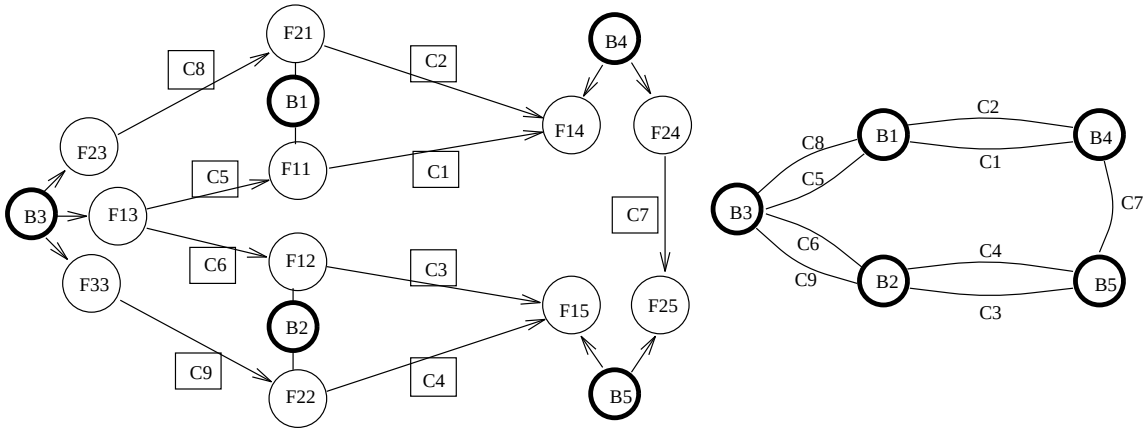

Figure 5.3 Spatial Constraint Graph for Cartesian Table

Table 5.2 Constraint Graph Basic Cycles

Cycle Name	*Cycle Equations*
$C_1 - C_2$	$F_{11}.C_1 = F_{21}.C_2$
$C_3 - C_4$	$F_{12}.C_3 = F_{22}.C_4$
$C_6 - C_9$	$F_{13}.C_6.F_{12}^{-1} = F_{33}.C_9.F_{22}^{-1}$
$C_8 - C_5$	$F_{23}.C_8.F_{21}^{-1} = F_{13}.C_5.F_{11}^{-1}$
$C_5 - C_1 - C_7 - C_3 - C_6$	$C_5.C_1.F_{14}^{-1}.F_{24}.C_7 = C_6.C_3.F_{15}^{-1}.F_{25}$

In Figure 5.1(f) the spanning tree corresponding to this example is shown, as well as the corresponding set of cords, c_i. It can be observed that every cord closes exactly one independent cycle when added to the spanning tree. The matrix equations describing the constraint chains for each cycle appear in Table 5.2.

At this point in the context of the Cartesian Table example, a partition of the original GCS/SF problem has been determined by using a basic set of cycles for the SC graph. This set of cycles serves the purpose of stating the complete and non-redundant set of simultaneous equations to be input to a Grobner Basis algorithm. In what follows, the canonical formulation for the Cartesian Table will be processed by the Brute Force, Divide & Conquer and Incremental Instancing approaches.

5.5.3.2 Brute Force Procedure

The Brute Force approach directly processes a polynomial set that contains all the cycle-matrix equations originated from the partition of the SC graph (Table 5.2). A lexicographically ordered Grobner Basis is then calculated for this complete set, with no calculation of partial or intermediate solutions. The Grobner Basis considering the order $S\phi_1 \succ C\phi_1 \succ y_1 \succ z_1 \succ S\theta_1 \succ C\theta_1 \succ S\phi_2 \succ C\phi_2 \succ y_2 \succ z_2 \succ S\theta_2 \succ C\theta_2 \succ S\phi_3 \succ C\phi_3 \succ y_3 \succ z_3 \succ S\theta_3 \succ C\theta_3 \succ S\phi_4 \succ C\phi_4 \succ y_4 \succ z_4 \succ S\theta_4 \succ C\theta_4 \succ S\theta_5 \succ C\theta_5 \succ x_5 \succ S\theta_6 \succ C\theta_6 \succ x_6 \succ S\theta_7 \succ C\theta_7 \succ y_7 \succ z_7 \succ S\theta_8 \succ C\theta_8 \succ x_8 \succ S\theta_9 \succ C\theta_9 \succ x_9$ is as follows:

$$\underline{S\phi_1} + S\theta_4\, C\phi_4 = 0 \tag{5.7}$$

$$\underline{C\phi_1} = 0$$

$$\underline{y_1} - S\theta_4\, z_4 - 3 = 0$$

$$5\,\underline{z_1} - 10\,S\theta_4 + y_4\,S\theta_4\,y_7 + 5\,S\theta_4\,y_4 - 2\,S\theta_4\,y_7 - C\phi_4\,z_7\,S\theta_4\,y_7 - 5\,C\phi_4\,z_7\,S\theta_4 = 0$$

$$5\,\underline{S\theta_1} - C\phi_4\,z_7 + y_4 - 2 = 0$$

$$\underline{C\theta_1} = 0$$

$$\underline{S\phi_2} = 0$$

$$\underline{C\phi_2} - S\theta_4\,S\theta_7 = 0$$

$$\underline{y_2} + S\theta_4\,y_7 + 5\,S\theta_4 - 2 = 0$$

$$5\,\underline{z_2} - S\theta_4\,z_4\,C\phi_4\,z_7 - 2\,C\phi_4\,z_7 + S\theta_4\,z_4\,y_4 + 2\,y_4 - 2\,S\theta_4\,z_4 - 4 = 0$$

$$\underline{S\theta_2} = 0$$

$$\underline{5}\,C\theta_2 + C\phi_4\,z_7 - y_4 + 2 = 0$$

$$\underline{S\phi_3} + S\theta_4\,C\phi_4 = 0$$

$$\underline{C\phi_3} = 0$$

$$\underline{y_3} - 1 - S\theta_4\,z_4 = 0$$

$$\underline{z_3} - 2\,S\theta_4 + S\theta_4\,y_4 = 0$$

$$\underline{S\theta_3} = 0$$

$$\underline{C\theta_3} + S\theta_4 = 0$$

$$\underline{S\phi_4} = 0$$

$$\underline{C\phi_4}^2 - 1 = 0$$

$$\underline{C\phi_4}\, y_4 - 2\, C\phi_4 + 5\, S\theta_7 - z_7 = 0$$

$$\underline{5\, C\phi_4\, S\theta_7} + y_4 - C\phi_4\, z_7 - 2 = 0$$

$$\underline{C\phi_4\, {z_7}^2} + 2\, z_7 - 25\, C\phi_4 - z_7\, y_4 - 5\, S\theta_7\, y_4 + 10\, S\theta_7 = 0$$

$$\underline{{y_4}^2} - 4\, y_4 - 21 + 10\, S\theta_7\, z_7 - {z_7}^2 = 0$$

$$\underline{{S\theta_4}^2} - 1 = 0$$

$$\underline{C\theta_4} = 0$$

$$\underline{S\theta_5} + 1 = 0$$

$$\underline{C\theta_5} = 0$$

$$\underline{x_5} - 1 = 0$$

$$\underline{S\theta_6} - 1 = 0$$

$$\underline{C\theta_6} = 0$$

$$\underline{x_6} + 1 = 0$$

$$\underline{{S\theta_7}^2} - 1 = 0$$

$$\underline{C\theta_7} = 0$$

$$\underline{S\theta_8} = 0$$

$$\underline{C\theta_8} - 1 = 0$$

$$\underline{x_8} - 2 = 0$$

$$\underline{S\theta_9} = 0$$

$$\underline{C\theta_9} - 1 = 0$$

$$\underline{x_9} - 2 = 0$$

This lexicographic Grobner Basis, presented as a triangular set, allows the evaluation of the Zero dimensionality properties for the polynomial ideal. By applying the method-

ology and algorithms developed in previous chapters the following conclusions can be drawn: (i) the ideal is not Zero-dimensional; (ii) the table is restricted to move in planar translation, $Tp(y_7, z_7)$, with two degrees of freedom, $y_7 and z_7$; and (iii) a careful examination of the example indicates that the remaining degree of freedom, z_4, indeed has a physical significance, since the sub-assembly $B_1 - B_2 - B_3$ still keeps one degree of freedom when all the other objects in the space are positioned. It can move *along* the line intersecting planes F_{15} and F_{14}. Although in real machine tool design such a degree of freedom is unrealistic, in this example it has the capability to demonstrate that confining the sub-assembly $B_1 - B_2 - B_3$ onto a plane F_{25} *is not* a necessary condition for the cartesian movement of the table. In more general terms, this rather fine detail demonstrates that a *formal* degree of freedom analysis is necessary in an increasingly computerized design environment. The problem is intuitively simple to a human being.

5.5.3.3 Divide & Conquer Procedure

This section shows the results of the preprocessing (Divide & Conquer) applied to the individual cycles presented in Table 5.2. Observing Figure 5.2 and considering the constraints in cycles $C_1 - C_2$, $C_3 - C_4$, $C_5 - C_8$ and $C_6 - C_9$, it is intuitively evident that the constraint intersections represented by those cycles are indeed reducible, and the resulting constraints should be as shown in Table 5.3. However, they cannot be reduced by the topological techniques introduced by Herve and/or Thomas & Torras [19, 36] because this mechanism involves non-trivial constraints. It will be shown here that the results in Table 5.3 can be obtained in a local preprocessing of the constraints by using Grobner Basis techniques. This preprocessing requires the application of the relations, established in this investigation, between the properties of the Grobner Bases and the solutions for the GCS/SF problem. The application of the Divide & Conquer strategy to the Cartesian Table problem follows.

Local Preprocessing. Cycle $C_1 - C_2$

The (non-trivial) constraints C_1 and C_2 are of the type $LN - ON - PLN$, with the feature lines F_{11} and F_{21} being non-colinear. The simultaneous enforcement of the two

Table 5.3 Topological Basic Cycle Reductions

Cycle	*Path 1*	*Path 2*	*Reduced Constraint*	*Defining Geometry*
$C_1 - C_2$	$C_1 = F_{11} - ON - F_{14}$	$C_2 = F_{21} - ON - F_{14}$	G_p	F_{14}
$C_3 - C_4$	$C_3 = F_{12} - ON - F_{15}$	$C_4 = F_{22} - ON - F_{15}$	G_p	F_{15}
$C_5 - C_8$	$C_5 = F_{13} - ON - F_{11}$	$C_8 = F_{23} - ON - F_{21}$	I_4	-
$C_6 - C_9$	$C_6 = F_{13} - ON - F_{12}$	$C_9 = F_{33} - ON - F_{22}$	I_4	-

constraints under such a geometric condition produces a (trivial) constraint of the type G_p, planar sliding. It is expected that the following procedure will confirm this intuitive conclusion.

By using the cycle equations shown in Table 5.2 for cycle $C_1 - C_2$, and the order $S\phi_1 \succ C\phi_1 \succ y_1 \succ z_1 \succ S\theta_1 \succ C\theta_1 \succ S\phi_2 \succ C\phi_2 \succ y_2 \succ z_2 \succ S\theta_2 \succ C\theta_2$, the lexicographic Grobner Basis resulted in:

$$\underline{S\phi_1} - C\theta_2\, C\phi_2 = 0 \tag{5.8}$$

$$\underline{C\phi_1} + C\theta_2\, S\phi_2 = 0$$

$$\underline{y_1} - 1 + C\theta_2\, z_2 = 0$$

$$\underline{z_1} + 2\, C\theta_2 - C\theta_2\, y_2 = 0$$

$$\underline{S\theta_1} + C\theta_2 = 0$$

$$\underline{C\theta_1} = 0$$

$$\underline{S\phi_2}^2 + C{\phi_2}^2 - 1 = 0$$

$$\underline{S\theta_2} = 0$$

$$\underline{C\theta_2}^2 - 1 = 0$$

Using the interpretation background developed in chapter 3, it can be seen that y_2, z_2 and $C\phi_2$ are free variables since they appear in no polynomial p as $head(p)$. Consistently, the result of this preprocessing indicates that two angular degrees of freedom θ_1 and θ_2 are fixed. The degrees of freedom can be extracted from path C_2, and they clearly represent the planar sliding $G_p(\phi_2, y_2, z_2)$.

Local Preprocessing. Cycle $C_3 - C_4$

From Table 5.2 and Figure 5.2 it can be determined that the cycle $C_3 - C_4$ presents an identical situation as cycle $C_1 - C_2$ does. By using the cycle equations shown in Table 5.2 for cycle $C_3 - C_4$, and the order $S\phi_3 \succ C\phi_3 \succ y_3 \succ z_3 \succ S\theta_3 \succ C\theta_3 \succ S\phi_4 \succ C\phi_4 \succ y_4 \succ z_4 \succ S\theta_4 \succ C\theta_4$, the following lexicographic Grobner Basis is calculated:

$$\begin{gathered}
\underline{S\phi_3} + S\theta_4\, C\phi_4 = 0 \qquad (5.9)\\
\underline{C\phi_3} - S\theta_4\, S\phi_4 = 0\\
\underline{y_3} - 1 - S\theta_4\, z_4 = 0\\
\underline{z_3} - 2\, S\theta_4 + S\theta_4\, y_4 = 0\\
\underline{S\theta_3} = 0\\
\underline{C\theta_3} + S\theta_4 = 0\\
\underline{S\phi_4}^2 + C\phi_4{}^2 - 1 = 0\\
\underline{S\theta_4}^2 - 1 = 0\\
\underline{C\theta_4}
\end{gathered}$$

From this triangular Grobner Basis one can see that the free variables are z_4, y_4 and $C\phi_4$; therefore, three degrees of freedom z_4, y_4, ϕ_4 are left in the (trivial) constraint $G_P(z_4, y_4, \phi_4)$. As in the previous case, the cycle could not be reduced by a topology-based re-writing strategy for trivial constraints.

Local Preprocessing. Cycle $C_5 - C_8$

The satisfaction of constraints C_5 and C_8 at the same time can be assimilated to the simultaneous positioning of two pegs into holes *whose axes are perpendicular*. This geometric condition suppresses all degrees of freedom of the cycle. As before, this conclusion is expected from the calculation of the Grobner Basis for the polynomials corresponding to this cycle. The ordering $x_5 \succ S\theta_5 \succ C\theta_5 \succ x_8 \succ S\theta_8 \succ C\theta_8$ leads to a (lexicographic)

Grobner Basis:

$$\begin{aligned} \underline{x_5} - 1 &= 0 \\ \underline{S\theta_5} + 1 &= 0 \\ \underline{C\theta_5} &= 0 \\ \underline{x_8} - 2 &= 0 \\ \underline{S\theta_8} &= 0 \\ \underline{C\theta_8} - 1 &= 0 \end{aligned} \tag{5.10}$$

In this case no free variables are left and, as a consequence, bodies B_1 and B_3 have their relative movement completely constrained.

Local Preprocessing. Cycle $C_6 - C_9$

As in the case of the cycle $C_5 - C_8$, it is expected that all movement between bodies B_3 and B_2 should be restricted. The ordering $x_6 \succ S\theta_6 \succ C\theta_6 \succ x_9 \succ S\theta_9 \succ C\theta_9$ produces the (lexicographic) Grobner Basis:

$$\begin{aligned} \underline{x_6} + 1 &= 0 \\ \underline{S\theta_6} - 1 &= 0 \\ \underline{C\theta_6} &= 0 \\ \underline{x_9} - 2 &= 0 \\ \underline{S\theta_9} &= 0 \\ \underline{C\theta_9} - 1 &= 0 \end{aligned} \tag{5.11}$$

Again in this case, the triangular presentation of the Grobner Basis helps to show the Zero-dimensionality of this ideal; therefore all the variables are instanced, and the body B_3 is rigidly attached to body B_2 in this cycle of the constraint graph.

Global Processing. Full Graph

In this section the (g_i) Grobner Bases already calculated for the individual cycles

$\{gb_{1-2}, gb_{3-4}, gb_{5-8}, gb_{6-9}\}$ are used towards the calculation of the Grobner Basis for the whole constraint graph.

The same variable order was used as for the full cycle, Brute Force approach; the Grobner Basis obtained is the same as in Equation 5.7. The Grobner Bases for the individual cycles replaced the original constraint equations in the calculation of the overall Grobner Basis, which would therefore include all the variables and constraints in the graph.

5.5.3.4 Incremental Instancing Procedure

According to the Incremental Instancing algorithm presented in previous sections, the sequence of cycles considered in the execution is presented in Table 5.4. Cycle $C_1 - C_2$ produces an instancing of variables $C\theta_2$, $S\theta_2$, $S\theta_1$ and $C\theta_1$. This result confirms the fact that, as mentioned before, two rotational degrees of freedom are lost in this cycle. Cycle $C_3 - C_4$ presents a similar situation for variables $S\theta_3$, $C\theta_3$, $S\theta_4$ and $C\theta_4$, and so on. Notice that, in general, the order in which the cycles are considered is significant if they share variables (line 8 of the Incremental Instancing algorithm). In that case, a variable instanced in a processed cycle would become a constant for the later stages of the algorithm. In this particular example, the first four cycles considered do not have variables in common among themselves. Therefore they do not influence each other. However, they share variables with, and therefore contribute to lower the computational burden of, the last cycle, $C_5 - C_1 - C_7 - C_3 - C_6$.

5.6 Summary

This chapter has presented an algorithm for the determination of a basic set of cycles for the Spatial Constraint graph of the GCS/SF problem. The SC graph is a formalism which allows the automatic expression of the GCS/SF problem and its set of governing equations. The set of basic cycles for the SC graph indicates the minimum set of cycle equations that contain all the information of the problem.

Table 5.4 Statistics for Incremental Instancing Execution

Subgraph	*Instanced Values*	*# Vars*	*Equations*	*GB Size*
$C_1 - C_2$	$C\theta_2 \to 1$ $S\theta_2 \to 0$ $S\theta_1 \to -1$ $C\theta_1 \to 0$	12	16	9
$C_3 - C_4$	$S\theta_3 \to 0$ $C\theta_3 \to 1$ $S\theta_4 \to -1$ $C\theta_4 \to 0$	12	16	9
$C_5 - C_8$	$x_5 \to 1$ $x_8 \to 2$ $S\theta_5 \to -1$ $C\theta_5 \to 0$ $S\theta_8 \to 0$ $C\theta_8 \to 1$	6	14	6
$C_6 - C_9$	$x_6 \to -1$ $x_9 \to 2$ $S\theta_6 \to 1$ $C\theta_6 \to 0$ $S\theta_9 \to 0$ $C\theta_9 \to 1$	6	14	6
Full Graph	$C\phi_1 \to 0$ $S\phi_1 \to 1$ $S\phi_2 \to 0$ $C\phi_2 \to 1$ $S\phi_3 \to 1$ $C\phi_3 \to 0$ $S\phi_4 \to 0$ $C\phi_4 \to 1$ $C\theta_7 \to 0$ $S\theta_7 \to -1$	20	65	20

The set of basic cycles of the SC graph induces a complete enumeration of the equations governing the GCS/SF problem, regardless of the methods used to solve them. Additionally, each cycle can be separately processed in order to detect local inconsistencies, solution spaces of low dimensionality, or subproblems with weak dependencies to the general GCS/SF problem. Therefore, the problem of the identification of *convenient* basic cycles becomes relevant. A convenient cycle, in this context, is the one whose equations represent an ideal of low dimensionality. In other words, it allows one to draw conclusions or detect inconsistencies in the configuration of the entities involved in the cycle. Since such cycles tend to be small, a desirable set of cycles contains as many small cycles as possible. The algorithm presented here to extract a basic set of cycles from an SC graph uses a heuristic strategy to limit the size of the cycles that results in a complexity of $O(VE^2 + EV^2)$. It should be stated that although other algorithms studied are able to find sets with smaller cycles, their computational expense is exponential and outweighs the advantages of the Divide & Conquer techniques.

The application of the algorithms discussed to the kinematic analysis of the Cartesian Table mechanism illustrates how a set of basic cycles of the SC graph, can be used in the solution of the problem in terms of sub-assemblies. Within the Divide & Conquer techniques, the method of Incremental Instancing proved to be more efficient for this particular example. Incremental Instancing not only solves the small subproblems separately, but, by partially instancing the set of variables, it further reduces the size of the problem. The evaluation of the convenience of D&C and II techniques in the context of diverse GCS/SF problems will be carried out in later chapters.

CHAPTER 6

Comparison of Solution Techniques for the GCS/SF Problem

The goal of this chapter is to evaluate the performance of different techniques in modeling and solving the GCS/SF Problem as the characteristics of the physical problem change. These characteristics involve the number of entities present in the world (two-body vs. multi-body systems), and the nature of the constraints (trivial vs. non-trivial) imposed upon them. A set of examples illustrating a full combination of types of physical problems against methods of modeling and methods of solution would involve a large number of situations. Since many of them are not significant from the point of view of their applicability, they will not be considered. This aspect will be developed in following sections.

Given a specific GCS/SF problem, the investigator must chose techniques for *modeling* and *solution*. In modeling the problem, the alternative of using canonical vs. non-canonical variables needs to be addressed. This choice has important implications for the solution process: canonical variables seem to be a smaller and physically more meaningful way to express the GCS/SF Problem. In this chapter, examples to test this statement are presented.

Once a modeling technique is chosen, the issue of the solution method arises. Several methods have been developed in this work, and the purpose of this chapter is to provide a comparison of their performance. They are represented in two main strategies:

(1) The solution of a large, non-structured set of polynomial equations modeled using either canonical or non canonical variables.

(2) The processing of local subproblems as part of the solution for the general GCS/SF problem and the consolidation of the partial solutions into the general one.

6.1 Design of Examples

To identify modeling and solution techniques appropriate for each physical situation of the GCS/SF problem, the following aspects have to be considered:

- *Physical* characteristics of the problem:
  Two Body vs. Multi Body Systems, and Trivial vs. Non-trivial Constraints between the entities.

- *Method* of modeling:
  Canonical vs. Non-canonical formulations.

- *Processing* method:
  Preprocessing (PP) vs. Non-preprocessing (NPP) techniques.

In Figure 6.1 the different combinations of the factors presented above are shown in a tree diagram. Branches which are trimmed are considered superfluous or of little applicability in common situations:

(1) Systems with trivial constraints are not discussed extensively due to the scarcity of applicable situations. In almost any case of multi-body or multiple constraint systems, the SC graph produces non-trivial interactions.

(2) In dealing with multi-body systems and non-trivial constraints, the use of non-canonical variables makes the computations expensive. Therefore no further effort was made to assess preprocessing techniques in multiple entity problems expressed with non-canonical formulation.

Two restrictions to the physical problems dealt with in this chapter are:

(1) In general, the number of entities involved in the GCS/SF problem is larger than two. Although examples using two-body scenarios are discussed, the emphasis in the modeling and solution of the GCS/SF problem will be directed to systems with multiple entities.

(2) Due to their applicability, this investigation focuses on *contact* constraints, which are mainly non-trivial. Although trivial constraints are not excluded altogether, this chapter will emphasize *non-trivial* cases.

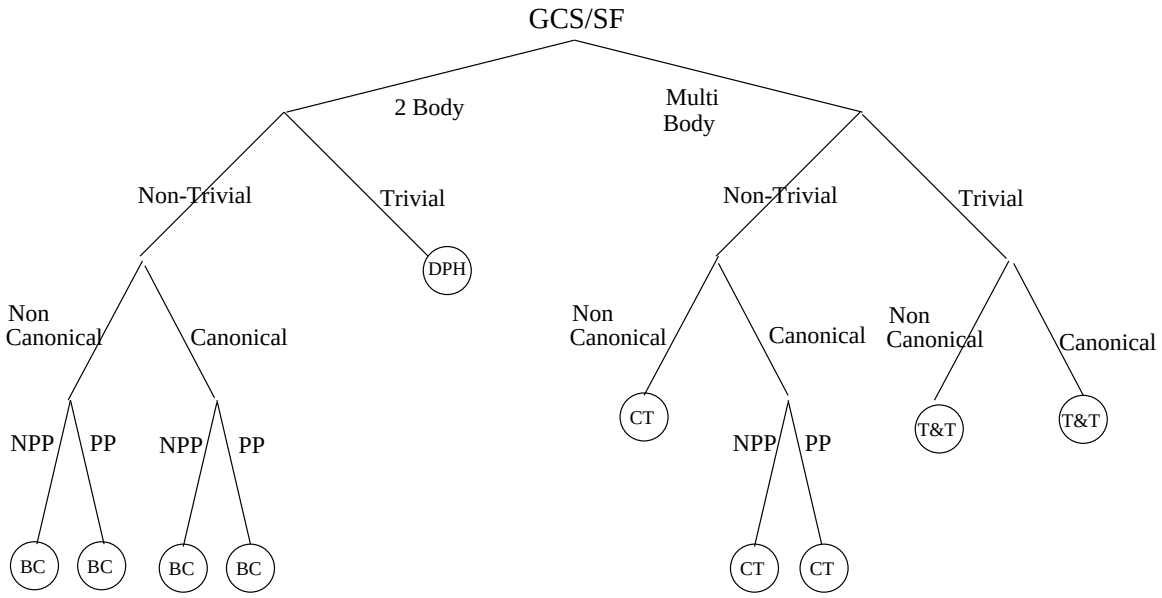

Figure 6.1 Layout of Examples

Figure 6.1 relates the examples presented later (BC, DPH, CT, T&T) with the characteristics and formulations of the GCS/SF problem. The first example, called **Double Peg and Hole (DPH)**, involves the simultaneous positioning of pegs into holes with parallel axes. DPH is used to illustrate the modeling and solution in the case of two-body systems with trivial constraints.

In order to explore the characteristics of systems with two bodies and multiple, non-trivial constraints, a configuration placing a block on a corner of the world is used.

Table 6.1 Entity Relations in Vector Form

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	$PLN=(p_2,n_2)$	$(p_1-p_2)\cdot n_2 = 0$
$LN-ON-LN$	$LN=(p_1,v_1)$	$LN=(p_2,v_2)$	$v_1\times v_2 = 0$ $(p_1-p_2)\times v_2 = 0$
$LN-ON-PLN$	$LN=(p_1,v_1)$	$PLN=(p_2,n_2)$	$(p_1-p_2)\cdot n_2 = 0$ $v_1\cdot n_2 = 0$
$PLN-ON-PLN$	$PLN=(p_1,n_1)$	$PLN=(p_2,n_2)$	$(p_1-p_2)\cdot n_2 = 0$ $n_1\cdot n_2 = \pm 1$

This example (called **Block on Corner (BC)**) presents a degree of redundancy in the constraints and is also suitable for application of preprocessing techniques.

An assembly adapted from [36] shows the interaction of more than two bodies with trivial constraints. It is referred to as **Thomas & Torras (T&T)** example. The case of multiple bodies with non-trivial constraints is illustrated by the modeling of a **Cartesian (or X-Y) Table (CT)**. Given the complexity of the SC graph, techniques for preprocessing become a necessity for this type of situations.

6.1.1 Terminology and Notation

In addition to the notation used in previous chapters, new terms are introduced here:

- O_{ij} is the origin of the feature frame i relative to entity B_j.
- X_{ij} is the orientation of x axis of the feature frame i relative to entity B_j.
- $B_j.O_{ij}$ is the absolute position of the point O_{ij}.
- $B_j.X_{ij}$ is the absolute orientation of the vector X_{ij}.
- C_i: Constraint Relations

  They are modeled following the formulations in Tables 6.1 and 6.2, repeated in this chapter for convenience.

Table 6.2 Entity Relations Using Canonical Formulation

Macro	*Joint Chain*	*Kinematic Joints in Chain*	*dof*
$P-ON-P$	S_o	spherical	3
$P-ON-LN$	$T_1 \circ S$	linear translation, spherical	4
$P-ON-PLN$	$T_2 \circ S$	planar translation, spherical	5
$LN-ON-LN$	C_v	cylindrical	2
$LN-ON-PLN$	$G_P \circ R_w$	planar sliding, revolute	4
$PLN-ON-PLN$	G_P	planar sliding	3

6.2 Two Body Systems

This section discusses the modeling and solution of GCS/SF problems in scenarios with two bodies. First, DPH will illustrate the case for trivial constraints. Second, BC will demonstrate the case when non-trivial constraints are present.

6.2.1 Trivial Constraints. Double Peg and Hole (DPH)

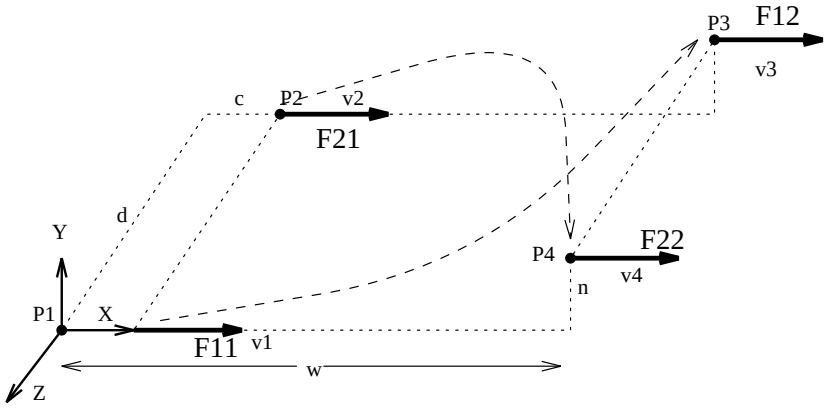

Figure 6.2 Double Peg and Hole (DPH) Example Scenario

Consider a scene in which there are two straight lines $F_{11} = (P_1, v_1)$ and $F_{21} = (P_2, v_2)$ (see Figure 6.2) expressed parametrically, and assumed to be rigidly linked to each other by a displacement M. Another set of lines, with similar conditions is given by $F_{12} = (P_3, v_3)$ and $F_{22} = (P_4, v_4)$. The proposed relations place $F_{11}\ ON\ F_{12}$ and $F_{21}\ ON\ F_{22}$ (F_{21}

Table 6.3 Constraints for the Double Peg and Hole (DPH) Example

Constraint	*Non-canonical Equations*	*Canonical Equations*
C1: $LN - ON - LN$ $F_{11} - ON - F_{12}$	$(B_1.X_{11} \times B_2.X_{12}) = 0$ $(B_1.O_{11} - B_2.O_{12}) \times (B_2.X_{12}) = 0$	$B_1.F_{11}.C_u.F_{12}^{-1} = B_2$
C2: $LN - ON - LN$ $F_{21} - ON - F_{22}$	$(B_1.X_{21}) \times (B_2.X_{22}) = 0$ $(B_1.O_{21} - B_2.O_{22}) \times (B_2.X_{22}) = 0$	$B_1.F_{21}.C_v.F_{22}^{-1} = B_2$

Table 6.4 Statistics for the DPH Example

Example	*Variable Type*	*Variables*	*Equations*	*GB Size*	*Time (secs)*
DPH	Non-canonical	12	20	16	1.53
DPH	Canonical	6	14	6	0.25

and F_{22} also being rigidly joined). Table 6.3 presents the canonical and non-canonical formulations for this example.

This problem presents a configuration similar to an electric outlet plug, with cylindrical legs. If considered separately, each peg-hole pair allows a cylindrical movement with axes F_{ij}'s. However, by joining them with a rigid link the rotational component of each pair is suppressed, leaving only the translational one in the direction of the lines F_{ij}. The statistics comparing the canonical and non-canonical modeling for this example are shown in Table 6.4. A detailed account of the modeling and solution of this case was presented in chapter 3.

6.2.2 Non-trivial Constraints. Block and Corner (BC)

An interesting situation for the application of non-canonical variables is an scenario with small number of entities, and a large number of (trivial or non-trivial) constraints among them. This situation originates in the fact that non-canonical modeling presents a number of variables proportional to the number of entities in the scene. Meanwhile, the number of variables for the canonical formulation is proportional to the number of constraints. Therefore, in this case, by using non-canonical modeling a smaller GCS/SF problem results. Since this example (Figure 6.3) presents large number of constraints

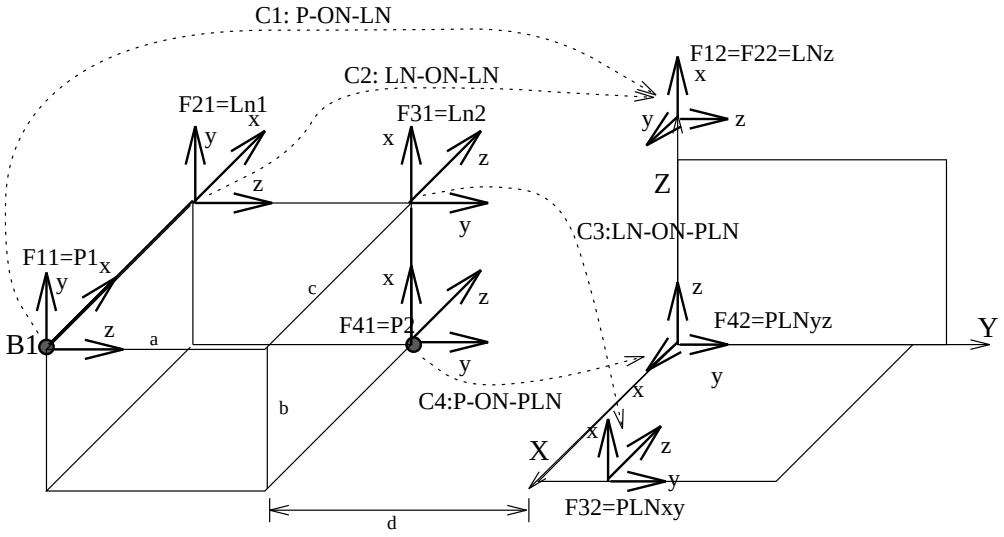

Figure 6.3 Block and Corner (BC) Example Scenario

with a low number of entities (B_1 and B_2), it is expected to show advantages in the non-canonical formulation.

The spatial relations specified in this example are shown in Table 6.5, column 1. The set of proposed constraints presents redundancies; for example constraint $C_2 : F_{21} - ON - F_{22}$ (*LN-ON-LN*) is stronger than $C_1 : F_{11} - ON - F_{12}$ (*P-ON-LN*). The general goal in this example is to place the block B_1 in the corner B_2.

Table 6.5 Constraints for the Block & Corner (BC) Example

Constraint	*Non-canonical Equations*	*Canonical Equations*
C1: $P - ON - PLN$ $F_{11} - ON - F_{12}$	$(B_1.O_{11} - B_2.O_{12}) \times (B_2.X_{12}) = 0$	$B_1.F_{11}.T_P.S_o = B_2.F_{12}$
C2: $LN - ON - LN$ $F_{21} - ON - F_{22}$	$(B_1.X_{21}) \times (B_2.X_{22}) = 0$ $(B_1.O_{21} - B_2.O_{22}) \times (B_2.X_{22}) = 0$	$B_1.F_{21}.C_v = B_2.F_{22}$
C3: $LN - ON - PLN$ $F_{31} - ON - F_{32}$	$(B_1.X_{31}).(B_2.X_{32}) = 0$ $(B_1.O_{31} - B_2.O_{32}).(B_2.X_{32}) = 0$	$B_1.F_{31}.R_u.G_P = B_2.F_{32}$
C4: $P - ON - PLN$ $F_{41} - ON - F_{42}$	$(B_1.O_{41} - B_2.O_{42}).(B_2.X_{42}) = 0$	$B_1.F_{41}.T_P.S_o = B_2.F_{42}$

Table 6.6 Canonical Formulation for Basic Cycles in the BC Example

Cycle Name	*Cycle Equations*
$C_1 - C_2$	$F_{11}.C_1F_{12}^{-1} = F_{21}.C_2.F_{22}^{-1}$
$C_2 - C_3$	$F_{21}.C2.F_{22}^{-1} = F_{31}.C3.F_{32}^{-1}$
$C_2 - C_4$	$F_{21}.C2.F_{22}^{-1} = F_{41}.C4.F_{42}^{-1}$

The set of equations of this example (Table 6.5) is dealt with by using both non-preprocessing and preprocessing techniques. In the first case (*NPP or Brute Force* techniques), the equations are directly included in a general, non-structured set. Their solution space is then calculated by the Grobner Basis methods discussed in previous chapters. In the second case, advantage is taken from the opportunity of applying the *Preprocessing (PP)* or *Divide and Conquer (D&C)* techniques based on the spatial constraint graph (Figure 6.4).

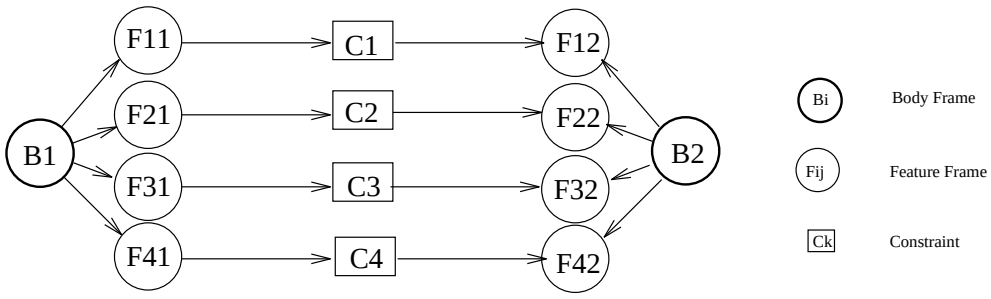

Figure 6.4 Constraint Graph for the BC System

The non-canonical and canonical formulations are presented in Table 6.5. Since a detailed discussion of these techniques was presented in chapters 3, 4 and 5, no further elaboration will be made here. Only the results of the different methods are shown in Table 6.7. For each case (canonical and non-canonical), the Brute Force approach uses the complete collection of constraints in the GCS/SF problem and determines the Grobner Basis for that set. In contrast, the D&C technique determines the equations originating in the basic cycles (Table 6.6) of the SC graph. These equations are required to calculate

Table 6.7 Statistics for the BC Example

Example	*Variable Type*	*Variables*	*Equations*	*GB Size*	*Time (secs)*
Brute Force	Non-canonical	12	18	12	2.334
$C1 - C2$	Non-canonical	12	15	10	2.10
$C2 - C3$	Non-canonical	12	14	11	2.25
$C2 - C4$	Non-canonical	12	13	11	1.916
Joint Cycles	Non-canonical	12	32	12	1.983
Total D&C	Non-canonical				8.249
Brute Force	Canonical	24	47	23	22.783
$C1 - C2$	Canonical	10	16	7	2.850
$C2 - C3$	Canonical	9	15	8	1.083
$C2 - C4$	Canonical	11	16	10	6.150
Joint Cycles	Canonical	24	25	23	6.916
Total D&C	Canonical				16.27

partial Grobner Bases GB_i. The GB_i bases are subsequently used in determining the Grobner Basis for the whole problem.

As expected, the non-canonical formulation exhibits an advantage with relation to the canonical one. However, D&C was largely unsuccessful. The added cost of calculating the cycle-based, local Grobner Bases along with calculating GB for the general set is higher than the direct approach of attacking the whole set of original equations. The small size of the problem makes the overhead of setting up the local subproblems more expensive than the solutions themselves. Therefore, in small systems the D&C approach is not desirable.

6.3 Multi-body Systems

Most of the instances of the GCS/SF problem correspond to multi-body systems, with trivial and non-trivial constraints. Therefore, it is important to find an efficient methodology for stating and solving such problems. The first study case presented here corresponds to a trivially constrained system with several bodies. It is a modification of an assembly presented in [36] (T&T). The second, non-trivially constrained, multi-body system corresponds to the modeling of a Cartesian (or X-Y) Table (CT).

Table 6.8 Constraints for the T&T Example

Constraint	*Non-canonical Equations*	*Canonical Equations*
C1: $LN-ON-LN$ $F_{21}-ON-F_{13}$	$(B_1.X_{21})\times(B_2.X_{12})=0$ $(B_1.O_{21}-B_3.O_{13})\times(B_3.X_{13})=0$	$B_1.F_{21}.C_u = B_3.F_{13}$
C2: $LN-ON-LN$ $F_{31}-ON-F_{12}$	$(B_1.X_{31})\times(B_2.X_{12})=0$ $(B_1.O_{31}-B_2.O_{12})\times(B_2.X_{12})=0$	$B_1.F_{31}.C_u = B_2.F_{12}$
C3: $LN-ON-LN$ $F_{23}-ON-F_{22}$	$(B_3.X_{23})\times(B_2.X_{22})=0$ $(B_3.O_{23}-B_2.O_{22})\times(B_2.X_{22})=0$	$B_3.F_{23}.C_u = B_2.F_{22}$
C4: $LN-ON-LN$ $F_{14}-ON-F_{32}$	$(B_4.X_{14})\times(B_2.X_{32})=0$ $(B_4.O_{14}-B_2.O_{32})\times(B_2.X_{32})=0$	$B_4.F_{14}.C_u = B_2.F_{32}$
C5: $LN-ON-LN$ $F_{33}-ON-F_{14}$	$(B_3.X_{33})\times(B_4.X_{14})=0$ $(B_3.O_{33}-B_4.O_{14})\times(B_4.X_{14})=0$	$B_3.F_{33}.C_u = B_4.F_{14}$
C6: $LN-ON-LN$ $F_{11}-ON-F_{14}$	$(B_1.X_{11})\times(B_4.X_{14})=0$ $(B_1.O_{11}-B_4.O_{14})\times(B_4.X_{14})=0$	$B_1.F_{11}.C_u = B_4.F_{14}$

Table 6.9 Canonical Formulation of Basic Cycles in the T&T Example

Cycle Name	*Cycle Equations*
$C_6-C_4-C_2$	$F_{11}.C6.C4.F_{32}^{-1} = F_{31}.C2.F_{12}^{-1}$
$C_6-C_5-C_1$	$F_{11}.C6.C5.F_{33}^{-1} = F_{21}.C1.F_{13}^{-1}$
$C_2-C_6-C_5-C_3$	$F_{31}.C2.F12^{-1}.F22.C3.F23^{-1} = F_{11}.C6.C5.F_{33}^{-1}$

6.3.1 Trivial Constraints. Torras & Thomas (T&T)

The assembly for this example is shown in Figure 6.5. Its constraint specification appears in Table 6.8, column 1. The two internal bodies B_2 and B_3 are allowed a cylindrical movement about each other using the axes F_{22} and F_{23} respectively. However, both degrees of freedom, the translation and the rotation are restricted by the slides F_{21} and F_{31} in the interior of body B_1. The remaining movement of the assembly B_2-B_3, the sliding granted by the features F_{21} and F_{11}, is avoided by the pin B_4 through the enforcement of constraint pairs (R_6, R_5) and (R_6, R_4). For the whole scenario, the only degrees of freedom left (rotation and translation) are associated with the pin B_4. The SC graph for this example is shown in Figure 6.6. The cycle decomposition of the SC graph used for the statement of equations and application of D&C techniques is shown in Table 6.9.

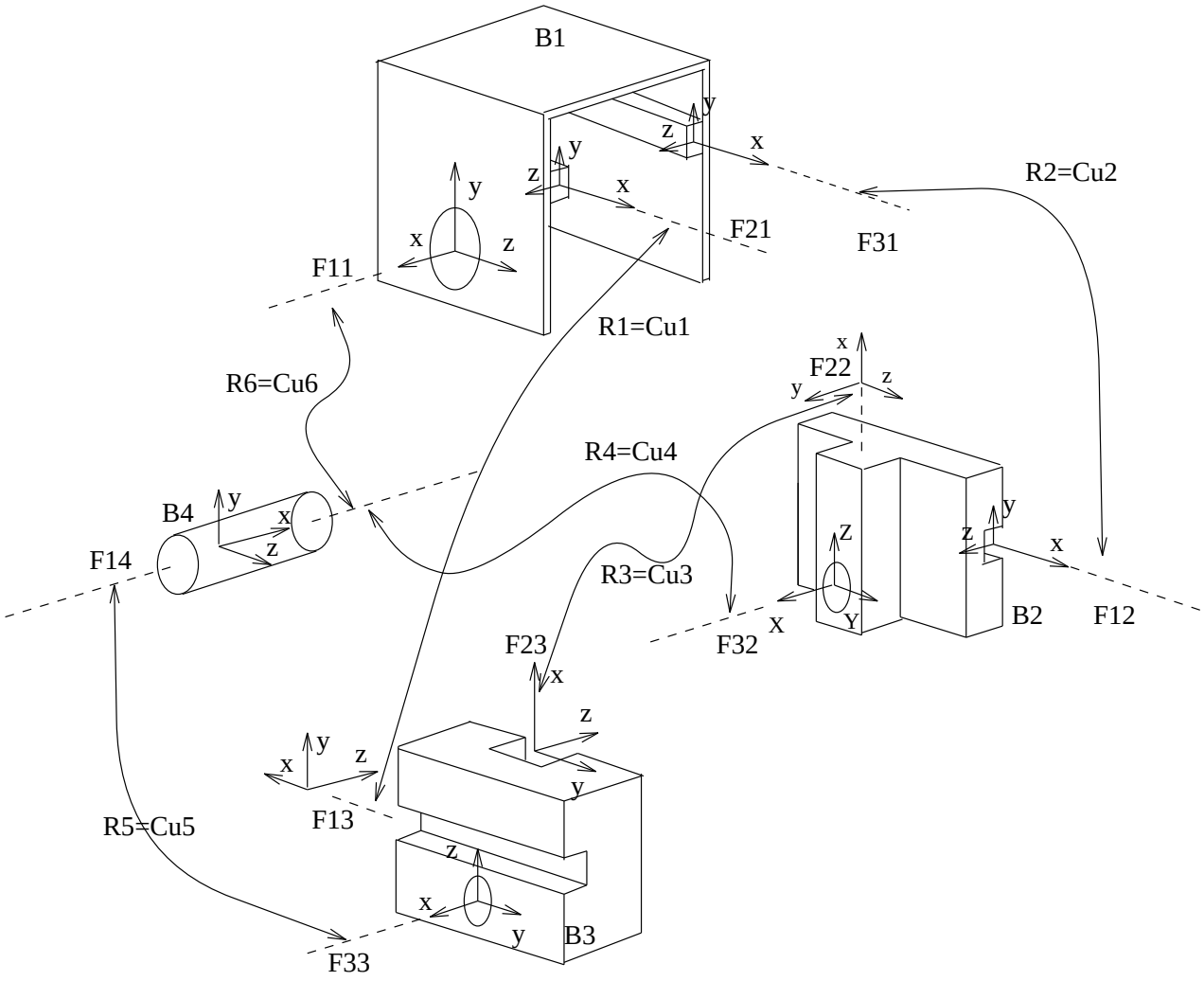

Figure 6.5 Torras & Thomas (T & T) Example Scenario

The non-canonical formulation for this example translates into the vector equations of Table 6.8, column 2. The canonical formulation appears in matrix form in column 3. The constraint graph includes three basic loops of constraints that must be simultaneously satisfied. Their expression in the form of matrix equations appears in Table 6.9.

Table 6.10 presents statistics for the T&T example. From these results, it can be concluded that in multibody, trivial constraint problems: (i) canonical formulation results in smaller polynomial sets, therefore lowering the computational expenses; (ii) for small problems (such as T&T), D&C refinements simply don't justify their own expense in problem formulation; and (iii) although D&C is comparatively more useful with non-

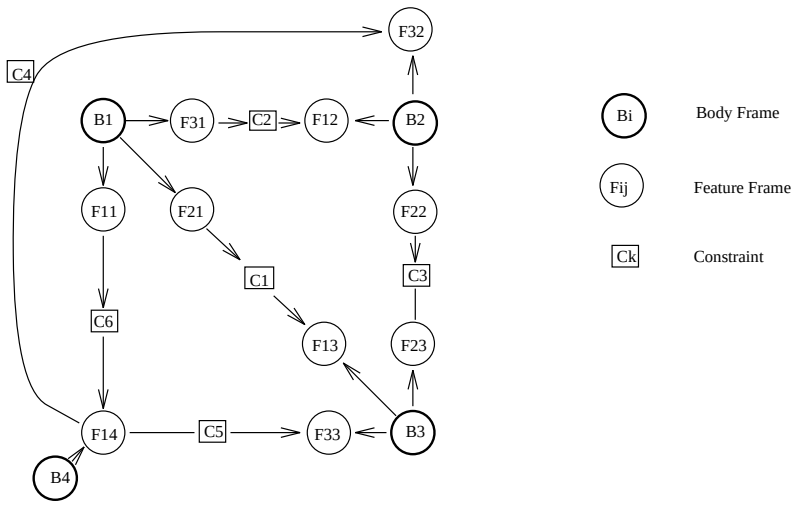

Figure 6.6 Constraint Graph for the T&T Example

canonical modeling because (ii), it simply becomes unpractical for large scale problems (with or without D&C). Even canonical formulation will solve the proposed problems only with the help of D&C techniques. The next example explores these statements in larger scenarios.

6.3.2 Non-trivial Constraints. Cartesian Table (CT)

In order to evaluate different modeling and solving techniques in multibody, non-trivial constraint systems, a mechanism called a Cartesian Table will be analyzed (see Figure 6.7). The account of constraints for this mechanism appears in Table 6.11, column 1. The spatial relations imposed in this example produce a constraint graph (with several cycles) which requires the application of D&C techniques for its solution. Although the comparison between non-canonical vs. canonical modeling is (tangentially) discussed in this example, the main purpose of it is to compare the application of Brute Force and D&C techniques. Local treatment of subproblems should offer advantages in lowering the computational resources expended. For the application of D&C techniques, it is necessary to extract a set of basic cycles in the constraint graph of Figure 6.8, as discussed in chapter 5. The result of the algorithms determining such a set of cycles is shown in Table 6.12.

Table 6.10 Statistics for the T&T Example.

Example	*Variable Type*	*Variables*	*Equations*	*GB Size*	*Time (secs)*
Brute Force	Non-canonical	36	54	37	403.3
$C1 - C5 - C6$	Non-canonical	24	30	25	117.0
$C2 - C4 - C6$	Non-canonical	24	30	25	117.1
$C2 - C6 - C5 - C3$	Non-canonical	36	16	-	∞
Joint Cycles	Non-canonical	36	56	37	20.400
Total D&C	Non-canonical				254.5
Brute Force	Canonical	18	42	16	5.800
$C1 - C5 - C6$	Canonical	9	15	7	1.400
$C2 - C4 - C6$	Canonical	9	15	7	1.600
$C2 - C6 - C5 - C3$	Canonical	12	16	12	4.817
Joint Cycles	Canonical	18	26	16	3.233
Total D&C	Canonical				11.05

Table 6.11 Constraints for the CT Example

Constraint	*Non-canonical Equations*	*Canonical Equations*
C1: $LN - ON - PLN$	$(B_1.X_{11}).(B_4.X_{14}) = 0$	$B_1.F_{11}.R_u.G_p = B_4.F_{14}$
$F_{11} - ON - F_{14}$	$(B_1.O_{11} - B_4.O_{14}).(B_4.X_{14}) = 0$	
C2: $LN - ON - PLN$	$(B_1.X_{21}).(B_4.X_{14}) = 0$	$B_1.F_{21}.R_u.G_p = B_4.F_{14}$
$F_{21} - ON - F_{14}$	$(B_1.O_{21} - B_4.O_{14}).(B_4.X_{14}) = 0$	
C3: $LN - ON - PLN$	$(B_2.X_{12}).(B_5.X_{15}) = 0$	$B_2.F_{12}.R_u.G_p = B_5.F_{15}$
$F_{12} - ON - F_{15}$	$(B_2.O_{12} - B_5.O_{15}).(B_5.X_{15}) = 0$	
C4: $LN - ON - PLN$	$(B_2.X_{22}).(B_5.X_{15}) = 0$	$B_2.F_{22}.R_u.G_p = B_5.F_{15}$
$F_{22} - ON - F_{15}$	$(B_2.O_{22} - B_5.O_{15}).(B_5.X_{15}) = 0$	
C5: $LN - ON - LN$	$(B_3.X_{13}) \times (B_1.X_{11}) = 0$	$B_3.F_{13}.C_u = B_1.F_{11}$
$F_{13} - ON - F_{11}$	$(B_3.O_{13} - B_1.O_{11}) \times (B_3.X_{13}) = 0$	
C6: $LN - ON - LN$	$(B_3.X_{13}) \times (B_2.X_{12}) = 0$	$B_3.F_{13}.C_u = B_2.F_{12}$
$F_{13} - ON - F_{12}$	$(B_3.O_{13} - B_2.O_{12}) \times (B_3.X_{13}) = 0$	
C7: $PLN - ON - PLN$	$(B_4.X_{24}) \times (B_5.X_{25}) = 0$	$B_4.F_{24}.G_p = B_5.F_{25}$
$F_{24} - ON - F_{25}$	$(B_4.O_{24} - B_5.O_{25}).(B_5.X_{25}) = 0$	
C8: $LN - ON - LN$	$(B_3.X_{23}) \times (B_1.X_{21}) = 0$	$B_3.F_{23}.C_u = B_1.F_{21}$
$F_{23} - ON - F_{21}$	$(B_3.O_{23} - B_1.O_{21}) \times (B_3.X_{23}) = 0$	
C9: $LN - ON - LN$	$(B_3.X_{33}) \times (B_2.X_{22}) = 0$	$B_3.F_{33}.C_u = B_2.F_{22}$
$F_{33} - ON - F_{22}$	$(B_3.O_{33} - B_2.O_{22}) \times (B_3.X_{33}) = 0$	

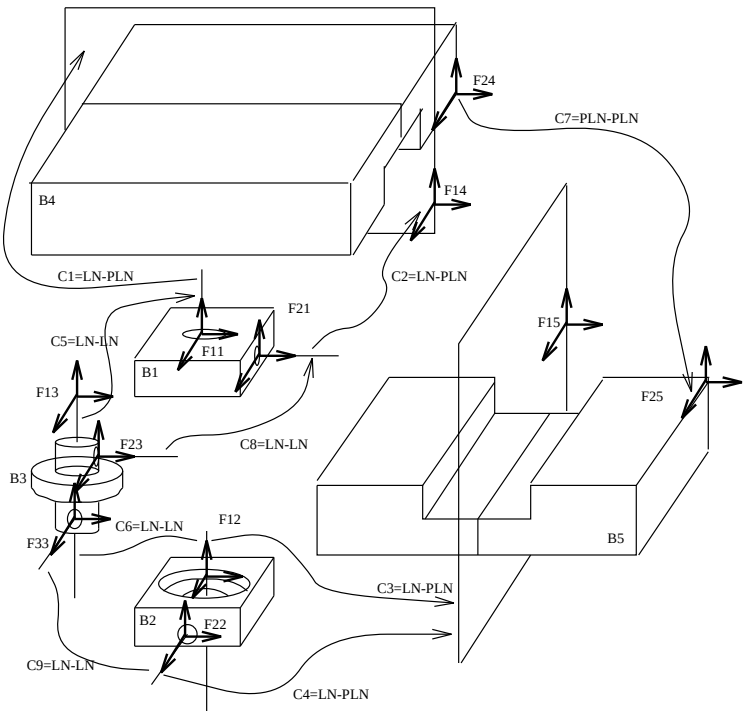

Figure 6.7 Cartesian Table (CT) Example Scenario

The CT example was initially modeled using non-canonical variables. This modeling was not successful since the execution calculating the Grobner Basis corresponding to the complete set of equations (Table 6.13) did not terminate. Therefore, pursuit of the modeling by non-canonical means was stopped, and the emphasis of this section was placed in evaluating variants using the canonical modeling. Using this formulation and a Brute Force (non-preprocessing) approach, the Grobner Basis was successfully determined. However, the high cost of this result forced the exploration of D&C techniques, presented next.

6.3.2.1 Divide & Conquer Preprocessing

As explained earlier, the *Divide & Conquer* approach uses the Grobner Bases corresponding to the local GCS/SF subproblems $GB_i = GB(S_i)$ instead of the original subproblem equations S_i. The theoretical justification for this substitution lies in the

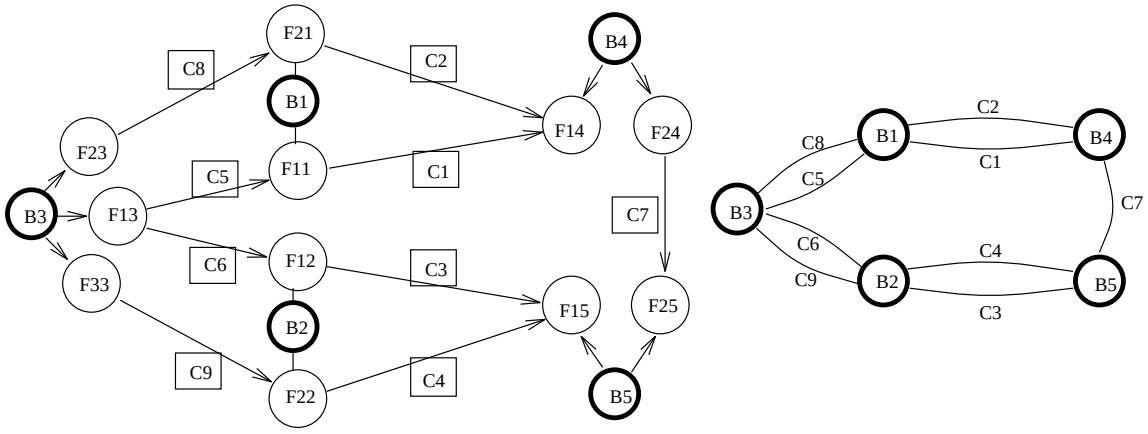

Figure 6.8 Constraint Graph for the CT Example

Table 6.12 Basic Cycles in the CT Example

Cycle Name	*Cycle Equations*
$C_1 - C_2$	$F_{11}.C_1 = F_{21}.C_2$
$C_3 - C_4$	$F_{12}.C_3 = F_{22}.C_4$
$C_6 - C_9$	$F_{13}.C_6.F_{12}^{-1} = F_{33}.C_9.F_{22}^{-1}$
$C_8 - C_5$	$F_{23}.C_8.F_{21}^{-1} = F_{13}.C_5.F_{11}^{-1}$
$C_5 - C_1 - C_7 - C_3 - C_6$	$C_5.C_1.F_{14}^{-1}.F_{24}.C_7 = C_6.C_3.F_{15}^{-1}.F_{25}$

fact that the two sets, the original S_i for subproblem (or cycle) i and the corresponding basis GB_i generate the same ideal. Refer to chapter 5 for a closer analysis of the D&C techniques.

The set of basic cycles which resulted from the partition algorithm applied to the SC graph is (see Figure 6.8):

$$SBC = \{\{C_1 - C_2\}, \{C_3 - C_4\}, \{C_6 - C_9\}, \{C_8 - C_5\}, \{C_5 - C_1 - C_7 - C_3 - C_6\}\} \quad (6.1)$$

Since these cycles are formed by non-trivial constraints, direct application of the reduction technique described in [19] by Herve is not possible. However, by using the canonical formulation, in combination with the properties of the Grobner Bases and the cycle preprocessing, the reduced constraints of Table 6.14 were obtained. In two cases

Table 6.13 Statistics for the CT Example. Divide & Conquer Strategy

Example	*Variable Type*	*Variables*	*Equations*	*GB Size*	*Time (secs)*
Brute Force Full Graph	Non-canonical	48	58	-	∞
Brute Force Full Graph	Canonical	40	73	40	107.416
$C_1 - C_2$	Canonical	12	16	9	1.883
$C_3 - C_4$	Canonical	12	16	9	2.050
$C_5 - C_8$	Canonical	6	14	6	0.617
$C_6 - C_9$	Canonical	6	14	6	1.034
Full Graph	Canonical	40	43	40	54.333
Total D & C	Canonical				59.9

Table 6.14 Topological Basic Cycle Reductions for the CT example

Path 1	*Path 2*	*Reduced Constraint*	*Constraining Geometry*
$C_1 = F_{11} - ON - F_{14}$	$C_2 = F_{21} - ON - F_{14}$	G_p	F_{14}
$C_3 = F_{12} - ON - F_{15}$	$C_4 = F_{22} - ON - F_{15}$	G_p	F_{15}
$C_5 = F_{13} - ON - F_{11}$	$C_8 = F_{23} - ON - F_{21}$	I_4	-
$C_6 = F_{13} - ON - F_{12}$	$C_9 = F_{33} - ON - F_{22}$	I_4	-

(cycles $C_5 - C_8$ and $C_6 - C_9$), the enforcement of all the constraints in a cycle produced a complete instancing of all variables. This instancing implies that bodies B_1, B_2, and B_3 are rigidly linked together, and the role of this *cluster* body B_{123} is to eliminate from the constraint C_7 the rotational degree of freedom θ_7, thereby producing the cartesian movement sought.

6.3.2.2 Incremental Instancing

A variant of the D&C strategy, the *Incremental Instancing (II)* technique (see chapter 5) implies the progressive *elimination by instancing* of variables from the problem as a consequence of solutions found by calculating the local bases GB_i. This section examines this approach and reports its results.

According to the procedure defined in chapter 5 for Incremental Instancing, the variables instanced during the execution of the algorithm appear in Table 6.15. The comparison of the results of Incremental Instancing with those of D&C (Table 6.13) show

Table 6.15 Statistics for the CT Example. Incremental Instancing.

Subgraph	*Instanced Values*	*# Vars*	*Equations*	*GB size*	*Time (secs)*
$C_1 - C_2$	$C\theta_2 \rightarrow 1;\ S\theta_2 \rightarrow 0$ $S\theta_1 \rightarrow -1;\ C\theta_1 \rightarrow 0$	12	16	9	1.883
$C_3 - C_4$	$S\theta_3 \rightarrow 0;\ C\theta_3 \rightarrow 1$ $S\theta_4 \rightarrow -1;\ C\theta_4 \rightarrow 0$	12	16	9	2.200
$C_5 - C_8$	$x_5 \rightarrow 1;\ x_8 \rightarrow 2$ $S\theta_5 \rightarrow -1;\ C\theta_5 \rightarrow 0$ $S\theta_8 \rightarrow 0;\ C\theta_8 \rightarrow 1$	6	14	6	0.717
$C_6 - C_9$	$x_6 \rightarrow -1;\ x_9 \rightarrow 2$ $S\theta_6 \rightarrow 1;\ C\theta_6 \rightarrow 0$ $S\theta_9 \rightarrow 0;\ C\theta_9 \rightarrow 1$	6	14	6	0.717
Full Graph	$C\phi_1 \rightarrow 0;\ S\phi_1 \rightarrow 1$ $S\phi_2 \rightarrow 0;\ C\phi_2 \rightarrow 1$ $S\phi_3 \rightarrow 1;\ C\phi_3 \rightarrow 0$ $S\phi_4 \rightarrow 0;\ C\phi_4 \rightarrow 1$ $C\theta_7 \rightarrow 0;\ S\theta_7 \rightarrow -1$	20	65	20	10.23
Total Time					15.747

that the first method presents definite advantages. These advantages can be traced to the decreasing size of each subproblem being subsequently solved.

The examples developed showed that results in Table 6.14 can be obtained by a local preprocessing of the constraints using an algebraic method which: (i) identifies the degrees of freedom lost in local sub-problems; (ii) detects local geometric or topological inconsistencies; and (iii) reduces the size of the GCS/SF problem to the degrees of freedom left by the local instancing processes.

6.4 Summary

This chapter discussed the techniques most suitable to model and solve the different types of GCS/SF problems. Systems with only trivial constraints and two bodies are considered rare, as are systems with several bodies and only trivial constraints between them. For the rest of the cases, the following conclusions have been drawn from the experiments:

(1) Systems with a small number of bodies and large number of (possibly) redundant constraints between them are more effectively modeled by non-canonical methods. In either case (non-canonical or canonical), the use of D&C techniques does not reflect a definite improvement. A conjecture about this result lies in the fact that computations spent in problem set-up may be heavier than the actual Grobner Basis calculation for a small polynomial set.

(2) In systems with a larger number of bodies, canonical variables are definitely needed. The computation effort for the non-canonical version of the examples in most such cases is extremely heavy.

(3) For larger systems, the *Divide & Conquer* techniques are advisable, since they take advantage of the existence of sub-systems strongly constrained internally, and weakly related to the external world. These sub-systems correspond to cycles in the SC graph which have some of their degrees of freedom internally instanced. If D&C techniques are used, the local Grobner Bases are used in the solution of the general system. These GB_i sets are already ordered (lexicographically or by degree order) and free of redundancy and inconsistencies. Therefore, they contribute towards the computation of the final solution.

(4) *Incremental Instancing* presents the advantage of actually eliminating degrees of freedom from the variable set, therefore contributing to lower the computational expense of the solution. This technique, as demonstrated in the examples of the Cartesian Table, is frequently the only alternative in dealing with large systems. By exploiting the Incremental Instancing technique, the size of the entity set can be decreased due to clustering of several entities into one. Divide & Conquer (and Incremental Instancing) techniques have the characteristic that they are effective only in situations in which the burden of the original Brute Force approach is high. If the collateral burden of the numerical solution of triangular subsystems and other book-keeping activities is large compared to the size of the problem, the Divide & Conquer techniques are not justified.

This chapter has shown that problems with large numbers of entities and constraints can be modeled in a more efficient way using the canonical formulation. Not only does it produce smaller and more compact sets of polynomials, but the results of the Grobner Basis can be easily related to the physical degrees of freedom of the entities. For large problems, D&C becomes a necessary tool for solution. Since in any case a partition of the GCS/SF problem is required in order to establish the complete, non-redundant set of equations for the problem, the D&C (and Incremental Instancing) techniques can benefit from such calculation.

CHAPTER 7

Applications

The formulation and solution of the GCS/SF problem were discussed previously in this thesis. Canonical variables and Divide & Conquer strategies were presented as means to make the tasks of formulation and solution more efficient and physically meaningful. This chapter addresses the applicability of the concepts developed. First, it presents the services provided by the Static Geometric Reasoning library to a client program for Feature Extraction. The scope and usability of these services are discussed. The second application corresponds to Kinematic Analysis of Mechanisms. The proposed algorithms for constraint management are used to analyze the Oldham coupling [15]. The characteristics of Grobner Bases for the GCS/SF problem are mapped into variations of the kinematic structure of the mechanism. As third case study, the theory developed for evaluation of the solution space for the GCS/SF problem is applied to Mobility Analysis. The study case is the Bennett mechanism [4], a classical example of solution space topology being dependent on problem geometry. These three cases, in addition to the other examples in this document, demonstrate the relevance and applicability of this investigation.

7.1 Geometric Reasoning for a Feature Extraction Application

The Static Geometric Reasoning library has been used to serve a client application in Automatic Feature Recognition. Traditional solution approaches to the problem of

Feature Recognition formulate general algorithms for the identification of features from the solid model of a part. This approach has not been successful because the hard coded algorithms make underlying functional assumptions about the definition of the feature. Therefore, they have the following negative effects: (i) the process may identify as features sets of entities whose functionality is completely unrelated; and (ii) it may fail in identifying evident features, which could be extracted by using a more customized algorithm. Such an algorithm could apply the expert knowledge of a user, who could devise more efficient, although less general, identification strategies. This approach would avoid the interaction with large search spaces, therefore making the recognition computationally feasible.

In order to avoid these drawbacks, Marin & Ferreira [31] have proposed to take the feature recognition process a step early. Instead of writing hard coded algorithms which are intended to recognize features in all the cases and situations, they designed a a Feature Declaration and Recognition language which allows for the *automatic production* of recognition algorithms. Although automatic, this process produces highly customized strategies for recognition of the predominant features in a given manufacturing environment. Instead of being forced to use existing feature definitions, the user is equipped with the tools to (i) *declare* features; (ii) *automatically produce* recognition code; and (iii) *control* the efficiency of the algorithm by using the semantic context related to the feature in pruning the search space. The code thus produced can be attached to the model of a part, in the same way in which tolerance or dimension specifications are. It can be used at any required stage in the process of design / manufacture.

The feature definitions written in this language are compiled into a C++ program and linked with the Static Geometric Reasoning library. A brief description of the characteristics of the language and its interaction with the Geometric Reasoning library follows.

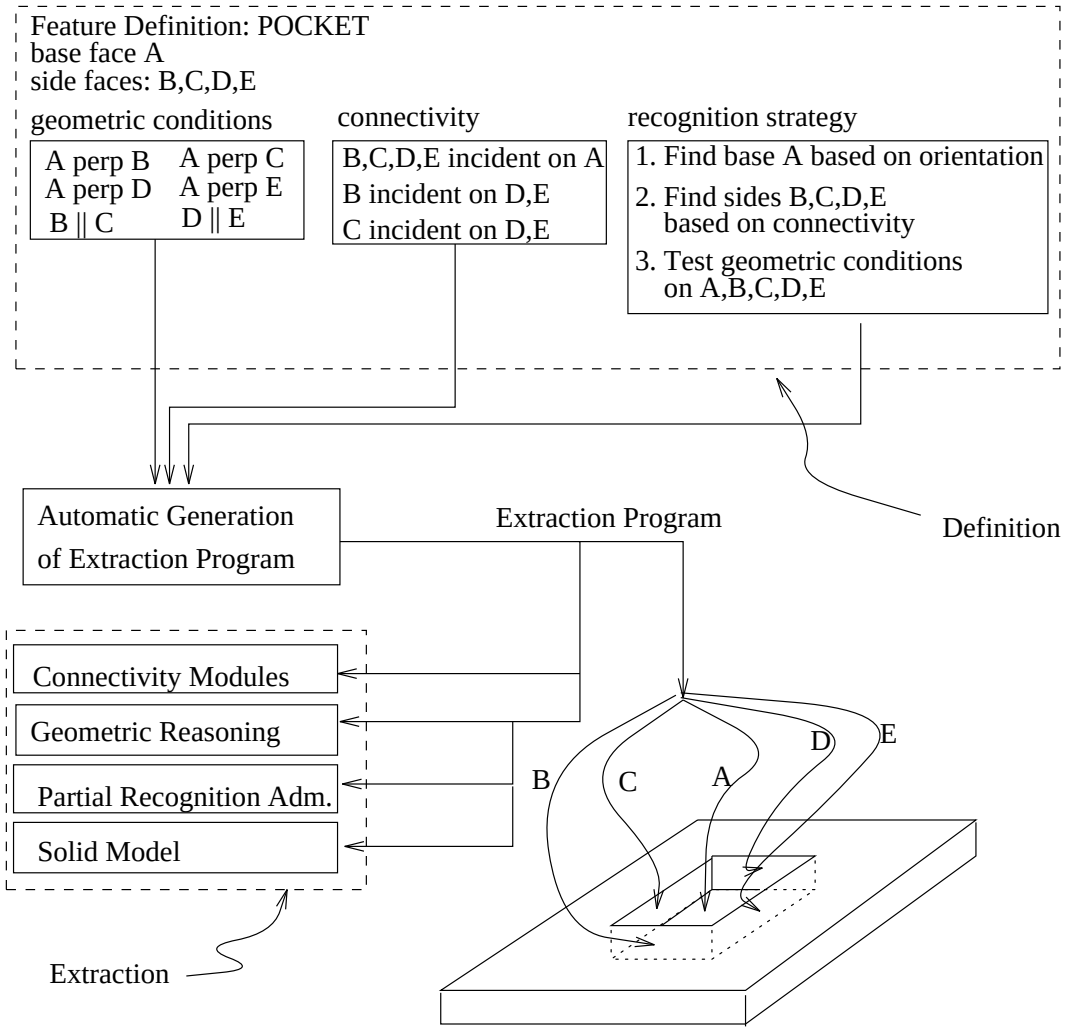

Figure 7.1 Feature Definition and Extraction System

7.1.1 Structure of the Reconfigurable Feature Definition and Extraction Application

The Reconfigurable Feature Definition and Extraction (RFDE) system is conceptually divided into two parts according to the service that it renders. The user interacts with the system in two stages:

(1) In the first stage the feature and its recognition strategy are defined and compiled into C++ routines. Figure 7.1 shows different aspects of the declaration and automatic routine generation processes.

(2) In the second stage, the extraction programs produced in the first one are executed when needed, using as input the solid model of the part. At this stage, the Extraction program makes extensive use of geometric and connectivity reasoning, administration of partial results, graphic display, etc.

To illustrate the above description, an example [31] is displayed and discussed. The feature to be recognized is a *step*, and its declaration / recognition program is transcribed here. The program is divided into *definition* and *recognition* sections. The definition section allows the user to establish the entities that constitute the feature, their properties, and the relations among them. The recognition part allows the expertise of the user to be expressed in the sequence of steps taken to identify the different entities which are components of the feature. These aspects are discussed next.

```
#Definition section
    0 DEFINITION
    1   NAME STEP ;
    2   ENTITIES:
    3       F1, F2 : FACE ;
    4       E1 : EDGE ;
    5   EDGE TYPE:
    6       E1 : STRAIGHT ;
    7   FACE TYPE:
    8       F1, F2 : PLANAR ;
    9   CONNECTIVITY:
    10      F1: (F2, E1, CONCAVE) ;
    11  ORIENTATION:
    12      F1, F2 : PERPENDICULAR ;
    13 END DEFINITION
#Recognition section
    14 BEGIN
    15  find [E1] based on edge convexity ;
    16  find [F1] based on connectivity with [F2] ;
    17  verify [E1] derived connectivity from [F1, F2]
    18 END
```

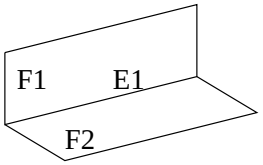

Figure 7.2 Simple step feature

7.1.2 Services Provided by the Geometric Reasoning System

Based on the example displayed here, the services provided by the Static Geometric Reasoning and other modules to the Feature Recognition module follow:

- *type of entities*:

  This call to the extraction routines is used to identify and extract entities by type (EDGE, FACE, STRAIGHT, etc) from the solid model. The entity types are declared in lines 2-4. They are (implicitly) enforced in lines 15-17.

- *characteristics of entities*:

  In this case, a geometric characteristic of the extracted entity is imposed; for example, the EDGEs are declared to be STRAIGHT, or the FACEs to be PLANAR (lines 5-8). The enforcement of these conditions is implicit in lines 15-17.

- *connectivity*:

  This statement identifies topological connections between entities in the solid model. In the example (lines 9-10), it is declared that the (FACE) entities F_1 and F_2 are connected by an EDGE (E_1), which should be concave. The Geometric Reasoning server produces the list of FACE entities connected to F_1. Enforcement of concavity on the common EDGE results from the compilation of line 15.

- *orientation*:

  Several orientation conditions (lines 11-12) may be requested between different entities. For example an EDGE perpendicular to another EDGE, two FACEs being parallel, etc. The role of the server is to test such relations in candidate sets

submitted to it. This conditions are implicitly used to reduce the search space (line 16).

From the discussion above it is clear that the language of the Reconfigurable Feature Definition and Recognition application is designed to have the *description* and *recognition* parts complement each other. Capabilities imparted to the language for feature description are met by corresponding types of enforcement statements that command actions on the recognition process. In this way, the language allows flexibility in the semantics associated to a feature name. In addition, the recognition process can be made more efficient by allowing the expertise of the user to dictate the order of enforcement of the descriptions.

7.1.3 Table Manager

From the point of view of the Feature Recognition process a *Feature* is an ordered set of n entities, or $n-tuple$, where each place of the tuple corresponds to a role specified in the recognitions program. In the example shown in Figure 7.2 a typical tuple would have the form (F_1, F_2, E_1). F_1, F_2 and E_1 have to be considered as *roles* which are to be fulfilled by real instances of entities. Therefore, the *result* of the recognition process is a set of tuples, each one of them holding different instances of entities from the solid model for the declared roles. This form of the result suggests that the administration of the set of results of the recognition process can be managed in similar way to a relational database. The manager of the data base, called Tuple Manager, handles the instancing of the roles in the tuples by real entities, deletes tuples which do not satisfy the requirements ($verify$ statements), and finishes the process returning a set of tuples, with their roles completely instanced. These tuples represent different combinations of entities in the solid model which satisfy the Feature Declaration. The recognition program fills up each tuple in the result set with pointers to the candidate entities in the model, as shown in Figure 7.1. The Tuple Manager is given the name *Partial Recognition Administration*,

since it manipulates the partially filled tuples until the recognition process finishes. At that point each surviving tuple is completely and consistently filled.

Figure 7.1 illustrates the modules serving the client program. They include (i) the Connectivity module, which serves the topological queries searching in the solid model; (ii) Geometric Reasoning module, which verifies geometric relations such as *perpendicular(), parallel()*, etc.; and (iii) the Tuple Manager, which administers the consistency and evolution of the Tuple Table.

7.1.4 Feature Extraction Client Program. Summary

This section has shown the use of the Geometric Reasoning server by a client application, a Reconfigurable System for Feature Description and Extraction. Besides the purely geometric queries performed by the server, other services such as topological (connectivity) queries, table administration, graphic display, etc., are provided for this application. These services are also organized in form of libraries and separate modules to be used by a client (human or program).

The following sections of this chapter involve the application of the theoretical background developed for the Dynamic Reasoning problem. The applications presented here are in the areas of Kinematic Analysis of Mechanisms and Mobility Analysis. These domains are particularly suitable for the application of this work since a mechanism is, by definition, the materialization of geometric constraints imposed upon its constitutive links. As a complement, Mobility Analysis implies the inventory of degrees of freedom of entities. The following sections expand on these topics.

7.2 Kinematic Analysis. Oldham Coupling

The purpose of this section is to show how the kinematics of mechanisms can be expressed in terms of the GCS/SF problem, and how the methods of solution for GCS/SF can be applied to analyze the characteristics of kinematic chains. Specifically, the Oldham coupling and several variations of it will be discussed.

Table 7.1 Joint List of the Oldham Coupling

Joint	*Joint Type*	*Canonical Representation*
R_1	Tu	$trans(x_1)$
R_2	Tu	$trans(x_2)$
R_3	Ru	$twix(\theta_3)$
R_4	Ru	$twix(\theta_4)$

The relevance of the GCS/SF problem in the context of Kinematic Analysis of Mechanisms is obvious given that kinematic joints are precisely constraints on the relative position of entities. A solution in the context of the GCS/SF problem corresponds to a physically realizable configuration (or a set of configurations) in the domain of kinematics. Continuous regions of the solution space for the GCS/SF problem directly map to the possible motion (degrees of freedom) of the mechanism. This correspondence allows a two way analysis: (i) a given kinematic arrangement can be screened by looking at the solution space of its expression as a GCS/SF problem; and (ii) it is possible to infer (new) kinematic arrangements based on desired properties of the solution space.

7.2.1 Kinematics of the Oldham Coupling

The Oldham coupling is shown in Figure 7.3. This mechanism is designed to connect two parallel, non-colinear axes, allowing the transmission of rotational movement [15]. Using canonical variables, the types of joints present in this mechanism are modelled in Table 7.1

The ground frame is called B_0, and supports the Oldham coupling through the features F_{20} and F_{10} which are the parallel, non-colinear axes. The two central joints R_1 and R_2 are prismatic, with their grooves F_{12} and F_{22} being non-parallel. R_3 and R_4 represent the rotatory movements (input / output) that are to be transmitted by the coupling.

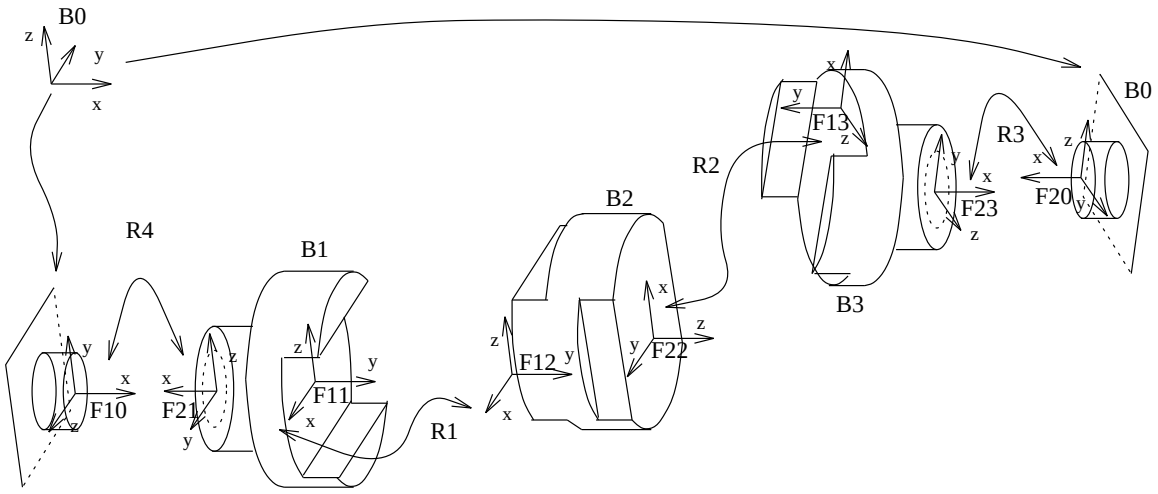

Figure 7.3 Piece Disassembly of Oldham Mechanism

7.2.1.1 Modeling of Oldham Coupling with Canonical Variables

Using the methodology developed in [29, 30] and the SC graph (Figure 7.4), the kinematic relations may be expressed by the following matrix equations:

$$B_1.F_{11}.R_1.F_{12}^{-1} = B_2 \tag{7.1}$$
$$B_2.F_{22}.R_2.F_{13}^{-1} = B_3$$
$$B_3.F_{23}.R_3.F_{20}^{-1} = B_0$$
$$B_0.F_{10}.R_4.F_{21}^{-1} = B_1$$

In this case the graph of constraints contains only one loop and it can be expressed by the equation:

$$F_{10}.R_4.F_{21}^{-1}.F_{11}.R_1.F_{12}^{-1}.F_{22}.R_2.F_{13}^{-1}.F_{23}.R_3.F_{20}^{-1} = I_4 \tag{7.2}$$

where I_4 is the 4×4 identity matrix. This equation conveys the topological configuration of the coupling.

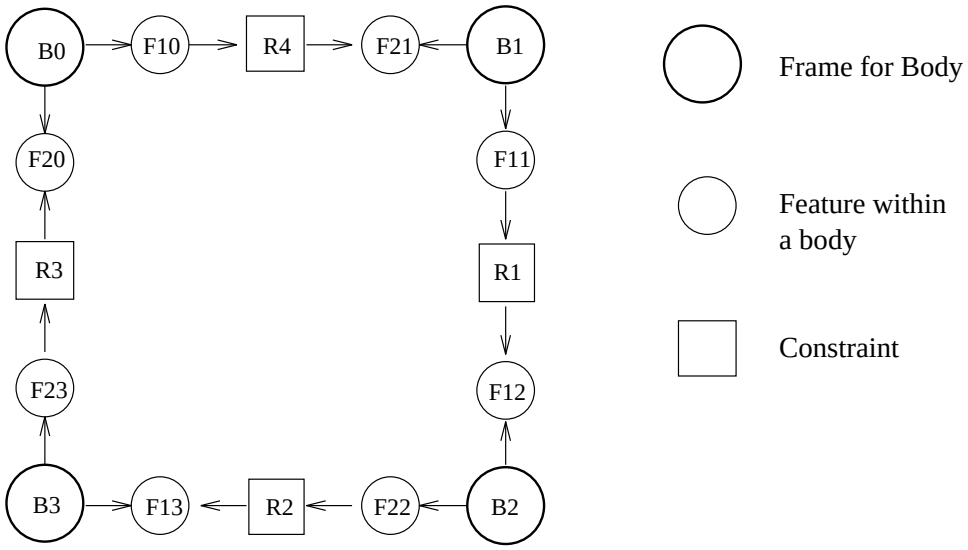

Figure 7.4 Graph of Spatial Constraints for Oldham Coupling

Figure 7.5 shows an abstraction of the Oldham coupling. It displays the dimensions L_i (the geometry) of the coupling. The non-colinearity between axes F_{20} and F_{10} is expressed by L_{13}. The total length of the mechanism is determined by parameter L_{12}. According to the equation 7.2 and the degrees of freedom of the joints as per Table 7.1, the polynomials which express the kinematics of the mechanism are:

$$-L_{12} + L_8 - L_{11} + L_7 + L_1 + L_3 = 0 \tag{7.3}$$

$$s\theta_4\, s\theta_3 - c\theta_4\, c\theta_3 - 1 = 0$$

$$-s\theta_4\, c\theta_3 - c\theta_4\, s\theta_3 = 0$$

$$L_{13}\, s\theta_4\, c\theta_3 + L_{13}\, c\theta_4\, s\theta_3 - s\theta_4\, x_2 + c\theta_4\, x_1 = 0$$

$$c\theta_4\, s\theta_3 + s\theta_4\, c\theta_3 = 0$$

$$s\theta_4\, s\theta_3 - c\theta_4\, c\theta_3 - 1 = 0$$

$$-L_{13}\, s\theta_4\, s\theta_3 + L_{13}\, c\theta_4\, c\theta_3 - c\theta_4\, x_2 - s\theta_4\, x_1 = 0$$

$$s\theta_3{}^2 + c\theta_3{}^2 - 1 = 0$$

$$s\theta_4{}^2 + c\theta_4{}^2 - 1 = 0$$

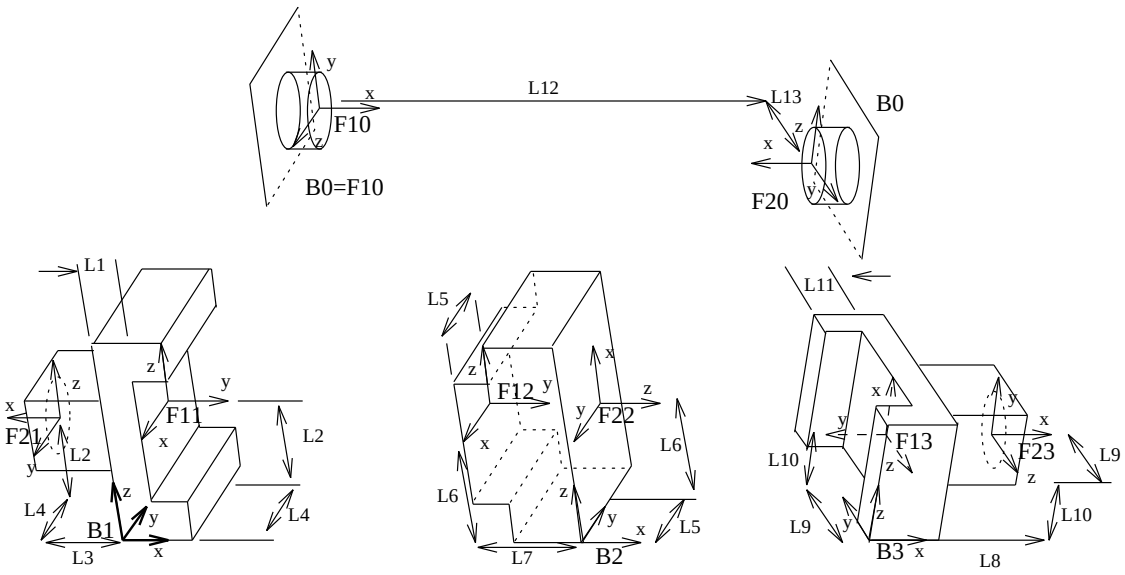

Figure 7.5 Dimensional Parameters for Oldham Coupling

Under the specification presented, the analysis of the Oldham coupling should predict the transmission of non-colinear, rotatory movement. In a first approximation, however, the Grobner Basis of this set of polynomials happens to be $GB = \{1\}$. This implies a topological or geometrical inconsistency. A careful examination of Equations 7.3 reveals that the inconsistency stems from the fact that in the present configuration the first equation in 7.3 cannot be made equal to zero if L_{12} is an *independent parameter*. Indeed, what this equation indicates is that

$$L_{12} = L_8 - L_{11} + L_7 + L_1 + L_3 \tag{7.4}$$

is a necessary condition for the mechanism to be realizable. This detection of geometrical inconsistencies in relation to the prescribed topology is a positive feature of using algebraic geometry techniques in the solution of the GCS/SF problem. The length L_{12} is therefore defined as in Equation 7.4. Under those conditions, the first equation in 7.3 becomes $0 = 0$. The lexicographic Grobner Basis, calculated under the order

$x_1 \succ x_2 \succ s\theta_3 \succ c\theta_3 \succ s\theta_4 \succ c\theta_4$, is:

$$\begin{aligned} \underline{x_1} + L_{13}\, s\theta_4 &= 0 \\ \underline{x_2} + L_{13}\, c\theta_4 &= 0 \\ \underline{s\theta_3} - s\theta_4 &= 0 \\ \underline{c\theta_3} + c\theta_4 &= 0 \\ \underline{s\theta_4}^2 + c\theta_4{}^2 - 1 &= 0 \end{aligned} \tag{7.5}$$

This triangularized Grobner Basis presents a free variable, $c\theta_4$, responsible for the one-dimensionality of the polynomial ideal [7, 22, 30]. It represents the rotational degree of freedom transmitted. As expected, the prismatic joints $R1$ and $R2$ (variables x_1 and x_2) are controlled by the separation between the two axes, L_{13}.

7.2.1.2 Variation 1. Cylindrical Joints in Central Connector

A question arising from the previous section is whether other configuration different from the Oldham coupling would also transfer rotational movement between non-colinear axes. One of such variations is achieved by replacing joints R_1 and R_2, which originally are prismatic (Tu), by cylindrical ones (Cu). Cylindrical joints present two degrees of freedom, ($Cu = trans(x, 0, 0).twix(\theta)$). The translational degree of freedom x is essential to the functioning of the coupling, and it would be expected that the rotational degree of freedom θ introduced be inoperant, since the nature of joints R_3 and R_4 forces a defined angular position in R_1 and R_2. This intuitive reasoning will be formally supported by the information contained in the Grobner Basis. Table 7.2 summarizes the degrees of freedom of the modified kinematic chain.

Using the cycle formulation from Equation 7.2, and the joint specification of Table 7.2, the kinematic equations can be formulated. Under the ordering $x_1 \succ s\theta_1 \succ c\theta_1 \succ x_2 \succ s\theta_2 \succ c\theta_2 \succ s\theta_3 \succ c\theta_3 \succ s\theta_4 \succ c\theta_4$ this equation set has the following lexicographic

Table 7.2 Joint List of the Oldham Coupling. Variation 1

Joint	*Joint Type*	*Canonical Representation*
R_1	Cu	$trans(x_1).twix(\theta_1\ )$
R_2	Cu	$trans(x_2).twix(\theta_1\ )$
R_3	Ru	$twix(\theta_3)$
R_4	Ru	$twix(\theta_4)$

Grobner Basis:

$$\underline{x_1} + L_{13}\, s\theta_4 = 0 \tag{7.6}$$

$$\underline{s\theta_1} = 0$$

$$\underline{c\theta_1} - 1 = 0$$

$$\underline{x_2} + L_{13}\, c\theta_4 = 0$$

$$\underline{s\theta_2} + 1 = 0$$

$$\underline{c\theta_2} = 0$$

$$\underline{s\theta_3} - s\theta_4 = 0$$

$$\underline{c\theta_3} + c\theta_4 = 0$$

$$\underline{s\theta_4}^2 + c\theta_4{}^2 - 1 = 0$$

The variable $c\theta_4$, the angular input/output of the mechanism, impedes the zero-dimensionality of the Grobner Basis. Therefore, it is effectively *the* degree of freedom left. All other variables appear as $head()$ of some polynomial. On the other hand, as expected, the angular freedoms given to the central joints R_1 and R_2 do not affect the degrees of freedom of the whole coupling, since they are immediately instanced. Therefore, the joints act as prismatic rather than cylindrical ones.

Table 7.3 Joint List of the Oldham Coupling. Variation 2

Joint	*Joint Type*	*Canonical Representation*
R_1	Tu	$trans(x_1)$
R_2	Tu	$trans(x_2)$
R_3	Cu	$trans(x_3).twix(\theta_3)$
R_4	Cu	$trans(x_4).twix(\theta_4)$

7.2.1.3 Variation 2. Cylindrical Joints in Input/Output Links

Under the condition of $R3$ and $R4$ being strictly rotational joints Ru, L_{12} was found dependent on other dimensions of the mechanism. This dependency suggests a variation in the basic Oldham coupling, namely the one obtained by allowing axial movement in joins $R3$ and $R4$. The expected result should be that the distance L_{12} can be considered an independent parameter and also that the coupling as a whole should act as a transmitter of cylindrical movement along two non-colinear parallel axes.

The constraint polynomials are built under the joint configuration of Table 7.3 and the matrix Equation 7.2. Their Grobner Basis under the ordering $x_1 \succ x_2 \succ x_3 \succ s\theta_3 \succ c\theta_3 \succ x_4 \succ s\theta_4 \succ c\theta_4$ is:

$$\begin{aligned}
\underline{x_1} + L_{13}\, s\theta_4 &= 0 \\
\underline{x_2} + L_{13}\, c\theta_4 &= 0 \\
\underline{x_3} + x_4 + L_{12} - L_8 + L_{11} - L_7 - L_1 - L_3 &= 0 \\
\underline{s\theta_3} - s\theta_4 &= 0 \\
\underline{c\theta_3} + c\theta_4 &= 0 \\
\underline{s\theta_4}^2 + c\theta_4{}^2 - 1 &= 0
\end{aligned} \tag{7.7}$$

This Grobner Basis represents a two-dimensional ideal with two free variables; x_4 and c_4. Variable c_4, as before, represents the rotational movement transmitted by the mechanism, while variable x_4 represents the translational degree of freedom. As predicted, the mechanism transmits cylindrical movement between non-colinear, parallel axes. Notice

that the variables x_4 and x_3 act as *slack* variables with regard to $L_{12}, L_8, L_{11}, L_7, L_1$ and L_3. This allows to have $L_{12} \neq L_8 - L_{11} + L_7 + L_1 + L_3$, in contrast with the original Oldham coupling in which such condition would render the mechanism unrealizable.

7.2.1.4 Variation 3. Parallel Prismatic Joints in Central Link

In this variation the *topology* of the Variant 2 is maintained while the *geometry* is modified in such a way that the solution space changes radically. The features F_{12} and F_{22}, the grooves of the prismatic joints R_1 and R_2, are made parallel. Since the original Oldham coupling was based on the fact that the link B_2 acted as an hypothenuse with sliding ends following the orthogonal features F_{12} and F_{22}, it is expected that this movement would be restricted if F_{12} and F_{22} are parallel. This effect would preclude the whole joint for transmitting rotational movement through the non-aligned axis F_{10} and F_{20}. Under the order $x_1 \succ x_2 \succ s\theta_3 \succ c\theta_3 \succ s\theta_4 \succ c\theta_4$ the lexicographic Grobner Basis of this arrangement would be:

$$\begin{aligned} \underline{x_1} + x_2 + L_{13}\, s\theta_4 &= 0 \\ \underline{s\theta_3} &= 0 \\ \underline{c\theta_3} + s\theta_4 &= 0 \\ \underline{s\theta_4}^2 - 1 &= 0 \\ \underline{c\theta_4} &= 0 \end{aligned} \tag{7.8}$$

This triangular Grobner Basis indicates that all the *angular* variables are locked (instanced), which means that the mechanism cannot transmit rotatory movement. As expected, x_2, the translational variable, is the only degree of freedom left. The freedom of x_2 implies that B_2 is able to slide, supported by B_1 and B_3.

7.2.2 Oldham Mechanism. Summary

It has been shown that the kinematics of mechanical link arrangements can be expressed in terms of the Geometric Constraint Satisfaction or Scene Feasibility (GCS/SF) problem. Therefore, the methods of solution for such a problem can be applied in order to analyze the characteristics of kinematic chains.

By using the set of canonical variables, a direct map between the kinematic characteristics of the mechanism and the algebraic properties of the Grobner Basis for the corresponding GCS/SF problem can be established. This direct map can be taken advantage of in inferring possible variations of the mechanism by elaborating about variations in the basis of the polynomial ideal. In the the GCS/SF problem a scenario was proposed, and the configuration of the solution space was found. The discussion presented in the previous sections shows that, constraints or specifications on the solution space can also be mapped back to the problem scenario in the form of conditions on the dimensions of the links. Other variations of the coupling have been presented to illustrate the statements above.

7.3 Mobility Analysis. The Bennett Mechanism

The Bennett mechanism, proposed in 1903 [4], is an example of a closed kinematic chain that in the general case behaves as a static structure, but presents mobility under certain lengths of its links and certain orientations of its joints. This mechanism is discussed to (i) demonstrate that Algebraic Geometry and Group Theory techniques can be used as tools for mobility analysis in problems which do not admit Thomas & Torras solution; and (ii) illustrate why a reasoning approach based on pure topology of the problem cannot succeed in the solution of the GCS/SF problem. In terms of the GCS/SF problem that is the object of this investigation, the Bennett mechanism is a classical case of the influence that the *geometry* of the scene has on the dimension of the solution space.

Among the general class of kinematic chains with multiple rotational joints, two variants are the ones that have parallel, no coincident axes (for example, the traditional four bar mechanism), and the ones whose axes intersect at one point (for example, the spherical joint). Let the class be of closed chains in which each link has rotational joints in the extremes, with arbitrarily oriented rotation axes on them. In general, such arrangements of links do not present mobility.

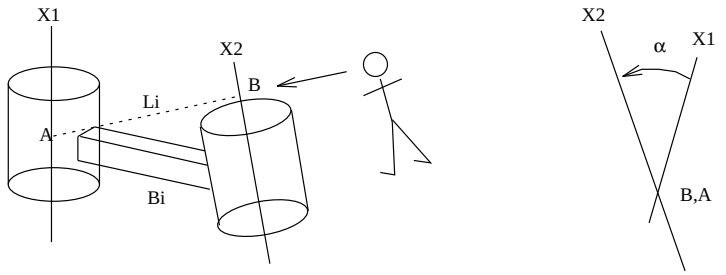

Figure 7.6 Basic Parameters of a Link

Each link can be abstracted as a central line perpendicular to two arbitrarily oriented rotation axes called X_1 and X_2 (Figure 7.6). Two parameters are needed to characterize this configuration: (i) the length L_i of the segment AB which is perpendicular to both axes; and (ii) the angle α_i between them. A disassembly of the mechanism proposed by Bennett is shown in Figure 7.7.

Bennett established the following sufficient conditions for the mobility of this mechanism: (i) pairs of links B_1-B_3 and B_2-B_4 must present identical parameters α_i and L_i; and (ii) the following relation must be satisfied:

$$\frac{sin(\alpha_1)}{L_1} = \frac{sin(\alpha_2)}{L_2} \tag{7.9}$$

This section analyzes a Bennett-compliant mechanism with the help of the canonical variables and the properties of Grobner Basis discussed in previous chapters. An example of a static structure, resulting from a four bar mechanism non-compliant with Bennett's

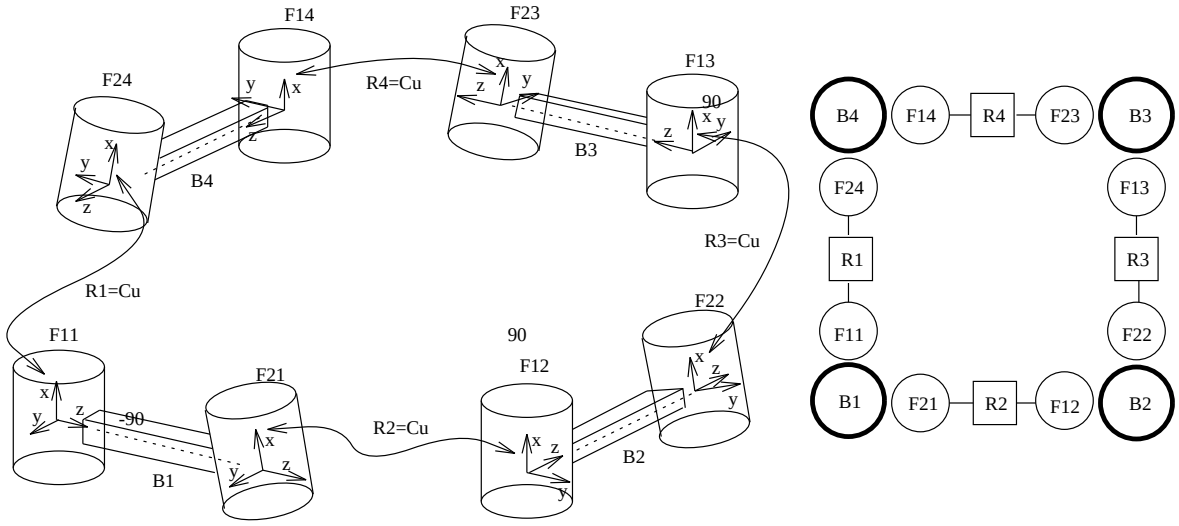

Figure 7.7 Bennett Mechanism

conditions, will also be examined. It illustrates a case in which departure from the Bennett conditions produces a *realizable* but yet rigid structure. Although Bennett's conditions ensure a mobile spatial kinematic four bar chain, the converse is not true. There could be realizable, (mobile) mechanisms which are not compliant with Bennett's conditions. One such case is discussed.

7.3.1 Modeling of Bennett-Compliant Mechanism

Figure 7.7 shows a Bennett-compliant mechanism. For simplification purposes, it has the central rod in every link B_k coincident with the segment perpendicular to the rotational axes of its joints F_{ik} $(i = 1, 2)$. If the conventions for assignment of frames are respected, the angle α is simply the angle between the X axes of the frames F_{1k} and F_{2k}. The cycle equation used to model this mechanism, as derived from the SC graph (Figure 7.7) is:

$$F_{21}.R_2.F_{12}^{-1}.F_{22}.R_3.F_{13}^{-1} = F_{11}.R_1.F_{24}^{-1}.F_{14}.R_4.F_{23}^{-1} \tag{7.10}$$

Its lexicographic Grobner Basis, calculated with the order: $s\theta_1 \succ c\theta_1 \succ s\theta_2 \succ c\theta_2 \succ s\theta_3 \succ c\theta_3 \succ s\theta_4 \succ c\theta_4$, resulted in:

$$\underline{s\theta_1} + s\theta_3 = 0$$
$$\underline{c\theta_1} - c\theta_3 = 0$$
$$\underline{s\theta_2} + s\theta_4 = 0$$
$$\underline{c\theta_2} - c\theta_4 = 0$$
$$\underline{s\theta_3}^2 + c\theta_3{}^2 - 1 = 0$$
$$s\theta_4\, s\theta_3 = 0$$
$$\underline{s\theta_3} + c\theta_4\, s\theta_3 = 0$$
$$\underline{s\theta_4} + s\theta_4\, c\theta_3 = 0$$
$$\underline{c\theta_3} + 1 + c\theta_3\, c\theta_4 + c\theta_4 = 0$$
$$\underline{c\theta_4}^2 + s\theta_4{}^2 - 1 = 0 \qquad (7.11)$$

In agreement with Bennett's findings, the mechanism presents one degree of freedom, represented by the variable $s\theta_4$. The fact that variable θ_4 is free, permits to use this mechanism to transmit rotatory movement between non-parallel axis. Notice that it presents more flexibility than the Oldham coupling, which required the axes to be parallel.

7.3.2 A Structure not Compliant with Bennett Conditions

The goal of this section is to find a physically realizable kinematic chain that is not mobile. In terms of the Grobner Basis, this represents a mechanism whose constraint equations produce a zero-dimensional ideal. Figure 7.8 shows a case of a four bar mechanism in which although the links B_1 and B_3 are identical (as well as the B_2 and B_4), the later ones present rotation axis X_1 *intersecting* X_2, therefore violating Bennett's sufficiency conditions. The cycle equations are identical to Equation 7.10. Specifying the

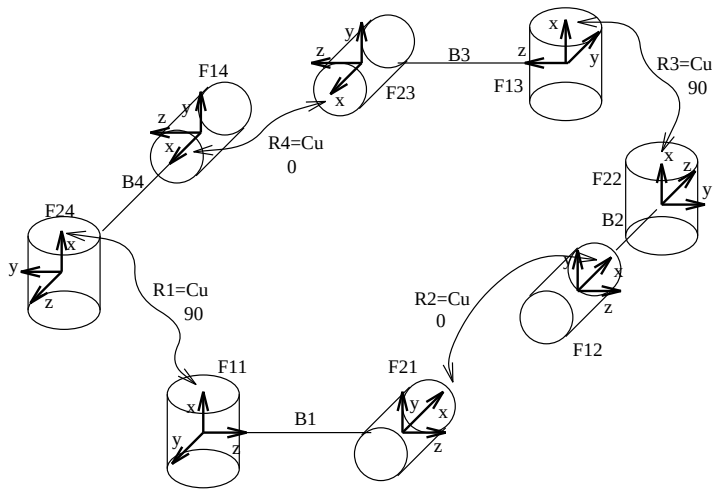

Figure 7.8 A static degeneration of a four bar mechanism

same lexicographic order as before, the following Grobner Basis is produced:

$$
\begin{aligned}
5\,\underline{s\theta_1} + c\theta_4 + 4 &= 0 \\
5\,\underline{c\theta_1} - 2\,c\theta_4 + 2 &= 0 \\
\underline{s\theta_2} &= 0 \\
\underline{c\theta_2} - c\theta_4 &= 0 \\
5\,\underline{s\theta_3} - c\theta_4 - 4 &= 0 \\
5\,\underline{c\theta_3} - 2\,c\theta_4 + 2 &= 0 \\
\underline{s\theta_4} &= 0 \\
\underline{c\theta_4}^2 - 1 &= 0
\end{aligned}
\tag{7.12}
$$

From the account of the *head* terms in the polynomials it is evident that this is indeed a zero-dimensional ideal, representing a kinematic chain physically realizable with zero degrees of freedom, therefore producing a static structure. This situation must be distinguished from the simpler case in which the kinematic chain is simply not realizable,

for example, due to dimensional incompatibility among the links, which would produce a Grobner Basis equal to $\{1\}$.

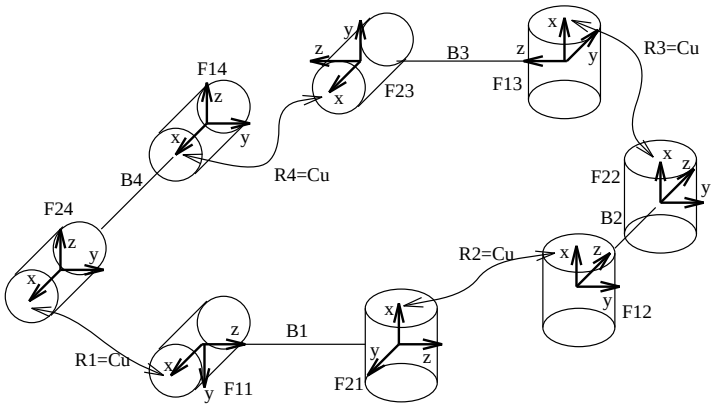

Figure 7.9 A variant of a four bar mechanism

7.3.3 A Mechanism not Compliant with Bennett Conditions

This section discusses a four bar mechanism, that demonstrates that Bennett's conditions are sufficient but not necessary; This mechanism does not comply with the Bennett conditions. Yet, it presents a degree of freedom.

The mechanism is shown in Figure 7.9. Its deviation from the Bennett's prescription lies in the coincident X axes of frames F_{14} and F_{24} in link B_4. The corresponding Grobner Basis is

$$\begin{aligned} \underline{S\theta_1} - S\theta_4 &= 0 \\ \underline{C\theta_1} + C\theta_4 &= 0 \\ \underline{S\theta_2} - 1 &= 0 \\ \underline{C\theta_2} &= 0 \\ \underline{S\theta_3} - 1 &= 0 \end{aligned} \tag{7.13}$$

$$\underline{C\theta_3} = 0$$

$$\underline{S\theta_4}^2 + C\theta_4{}^2 - 1 = 0$$

in which the variable $C\theta_4$ represents the remaining degree of freedom.

7.3.4 Bennett Mechanism. Summary

This example illustrated the following conclusions in the area of Mobility Analysis:

(1) The Grobner Basis analysis can be applied to a general case. Therefore, it complements methods (Thomas & Torras), which only apply to trivial constraints, and configurations which are topologically reducible. In this example, no reductions are possible, and however, the mobile mechanism presents a unique degree of freedom. These degrees of freedom are successfully identified by jointly applying the canonical formulation along with the relations between the Grobner Basis and the solution space for the GCS/SF problem.

(2) This application shows the need for the joint consideration of geometrical and topological conditions in determining the dimension of the solution space. In particular, Bennett conditions represent the *geometrical* information required to give mobility to a purely *topological* specification.

7.4 Summary

This chapter has shown additional applications of the techniques and algorithms developed in chapters 2, 3, 4, 5 and 6. The three areas addressed correspond to (i) application of the Static Reasoning Server to a client program for Reconfigurable Feature Definition and Extraction; (ii) application of Algebraic Geometry theory to Kinematic Analysis of Mechanisms. It demonstrates that specifications made in the solution space for existence of degrees of freedom can be mapped back to the physical domain of the GCS/SF; and (iii) Analysis of Mobility, applied to the Bennett Mechanism. It illustrates

the interdependency between topology and geometry in the configuration of a solution space for the GCS/SF problem.

CHAPTER 8

Conclusions and Recommendations

Application of geometric reasoning is central in CAD/CAM/CAPP environments. This investigation has undertaken the theoretical background, design and partial implementation of a Geometric Reasoning Server. This system is aimed at offering geometric manipulation and reasoning services to users (humans and computer programs). By using them, the user can access a series of routines and algorithms, which represent a large amount of research and work in the area of Computational Geometry applied to design and manufacture. The impact of such a set of routines is significant in industrial research and development environments, in which repetition of efforts is costly.

The existence of a Geometric Reasoning Library frees the user, a research or applications engineer, from the arduous work of programming a geometric reasoning algorithm. Such programming efforts include several steps: (i) understanding the central idea; (ii) mathematical formalization; (iii) development and formal test of a solution; (iv) design and implementation of supporting data structures, and of the algorithm itself; (v) debugging of topological and geometrical degeneracies; and (vi) display of the result, or communication of it through data structures to other programs or users.

The need for a library which supports reasoning in geometric scenarios in which the actual configuration is well defined (static reasoning) has been discussed in the initial chapters. However, an increasing number of applications in CAD / CAM / CAPP environments interact with partially defined scenarios (dynamic reasoning). Such situations occur when the scenario is a virtual world specified by relations among its components. Although in the static case there exist extremely hard problems in computational geom-

etry, the complexity of the problems in the dynamic case is much larger, starting with the evaluation of the *existence* of the scene itself.

This investigation has attacked the first problem by creating a geometric reasoning library which works with deterministic scenarios (static reasoning). The second type of problems has been addressed by identifying the theoretical basis for the characterization of uncertain scenarios (dynamic reasoning), by taking this theory to applicable methodologies and by showing its application in several domains. Uncertain scenarios are specified by geometric relations which translate into sets of polynomial equations. Techniques in Algebraic Geometry allow the characterization of the solution space of such sets of polynomials. A direct relation has been established in this work between the properties of polynomial ideals and the existence of feasible geometric "worlds".

In what follows, comments and conclusions relevant to specific areas of this work are presented. Finally, recommendations for future research are given, which signal the most promising areas of research. In addition, potentially difficult areas are identified.

8.1 Static Reasoning

A static reasoning library has been designed and developed, offering two types of services to human and program clients: (i) logical queries, which test the objects in the world for satisfaction of a given relation (parallelism, perpendicularity, inclusion, etc.); and (ii) construction queries, which create entities satisfying non-ambiguous relations with other ones in the existing world. Examples of this type of queries are the construction of lines perpendicular to other two lines, intersections of surfaces, lines, minimum distance between points and surfaces, etc. In addition to that, other entities are built whose mathematical specification may be very cumbersome, but whose algorithmic construction is well defined. Examples of this type of entities are the intersections between polygonal regions in 2D and 3D space, the convex hull enclosing a set of points in 2D and 3D space, the projections of lines and points, etc.

In addition to the purely geometrical tasks, the static reasoning system performs a variety of other roles. These roles are mediate the interaction of the user with the geometric world of objects, through the following modules:

(1) *naming and attribute managing*:

This module ensures the uniqueness of names for the objects in the world, and the consistency of object definition in cases where other objects in the world are deleted.

(2) *parsing and script interfacing*:

This module manages the interface language created for this application, which allows script- and user-driven interaction with the geometric world. It includes lexical and syntactic analyzers, and a uniform interface for user interaction.

(3) *object storage and retrieval*:

This module contains the data structures and classes of container objects which allow the client to organize, store and efficiently retrieve the entities in the world.

(4) *graphic interface*:

This module lends semantic effects to the subset of the interface language which deals with displaying tasks.

(5) *identification and extraction*:

This module extracts selected objects from more complex ones. It allows the flexible managing of entities resulting from geometric constructions whose *nature* and *number* are not known in advance.

8.1.1 An application of Static Reasoning

The static geometric reasoning library was used to service geometric and database requests from a Reconfigurable Feature Definition and Extraction client program. As described in chapter 7 (applications), this client program made use of the geometric reasoning services of a central kernel. These services covered tests for orientation, parallelism,

perpendicularity, metrics between the entities, etc. The Static Geometric Reasoning library also provided supporting routines assessing connectivity relations in the objects. The Feature Recognition client was served by an additional module functioning as data base administrator, which produced solution tables for the feature recognition program.

To close this subsection, it can be said that the static geometric reasoning kernel has proven the point for the convenience of the existence of a centralized server in computational geometry in CAD / CAM / CAPP tasks. Thanks to the emphasis placed on an open design, both in the algorithmics and in the interface, the server continues being expanded by other researchers, to include additional types of objects and services.

8.2 Dynamic Reasoning

Dynamic Reasoning covers the *Geometric Constraint Satisfaction or Scene Feasibility (GCS/SF)* problem. For convenience, the definition of the GCS/SF problem is repeated here: Let a World W be a closed, homogeneous subset of E^3, and a set of geometric entities $S = \{e_1, ..e_n\}$ which are closed, connected subsets of W. A set of spatial relations (or constraints) among pairs of entities $R = \{R_{i,j,k}\}$ is specified, where $R_{i,j,k}$ is the k^{th} relation between entities i and j. The solution to such a problem is constituted by either a diagnostic of *inconsistency* in the formulated relations, or an instance of a set of entities e_i in the world W consistent with all constraints R specified on entity i.

8.2.1 Algebraic Geometry Background

In this investigation the problem of reasoning about geometric constraints was addressed using Grobner Bases. A Grobner Basis of a polynomial set $F = \{p_1, p_2, ..p_n\}$ has several properties which characterize the variety of the polynomial ideal. From the GCS/SF problem perspective, these properties allow the user to determine: (i) if there are remaining spatial degrees of freedom among the entities in a given scenario or world (which would imply the existence of an infinite number of solution configurations); (ii) the redundancy of a constraint in the context of a pre-existent set of constraints; and (iii)

the (in)consistency of the set of constraints F, which would produce an empty solution space. The application of Grobner Bases to the GCS/SF problem allows the treatment of geometrical as well as topological inconsistencies in the constraint set. An algorithmic explanation of how the Grobner Bases properties can be exploited in a constraint-based scenario was proposed and applied in several examples.

8.2.2 Methodologies for Modeling

A drawback of the direct application of the Grobner Bases analysis technique is the growth of computational effort with problem size. Therefore, the set of variables used for modeling is an important consideration. The first alternative explored in this work for modeling of the GCS/SF problem was the use of position (non-canonical) variables for the specification of the entities in the world. Although theoretically sound, it produces a large problem formulation and computing expenses for the Grobner Bases calculation. This disadvantage was offset by the choice of a convenient set of (canonical) variables, dictated by the conjugation classes of the subgroups of the group $SE(3)$ of the Euclidean displacements. Canonical variables proved to be a compact representation of the GCS/SF problem constraints and to have a direct physical interpretation. Therefore, they facilitate the interpretation and analysis of the solution space of the constraints and the degrees of freedom of the entities involved.

The evaluation of the two methods proposed showed that systems with small numbers of bodies and large numbers of (possibly) redundant constraints between them are more effectively modeled by non-canonical methods, while systems with large numbers of bodies and few constraints in each pair of bodies are better modeled by canonical variables. The explanation for this behavior lies in the fact that non-canonical variables are *positional* while canonical ones are *motion-* or *freedom*-related. Since with a constant number of bodies the number of positional variables remains constant regardless of the number of constraints, this type of modeling is attractive for problems with small numbers of entities, and possibly large numbers of constraints between them. In problems

with large numbers of entities, the usage of their degrees of freedom instead of their positional variables produces a physically meaningful and compact problem formulation.

8.2.3 Methodologies for Solution

In spite of the improvement in performance achieved by the use of canonical formulation of the GCS/SF problem, additional efforts were made to find opportunities to lower the computational burden of solving this problem. The Divide & Conquer (D&C) techniques, discussed next, illustrate such efforts.

8.2.3.1 Divide & Conquer Techniques

Divide & Conquer techniques take advantage of strongly constrained subproblems which can be efficiently solved given their small size. Once these subsystems have been identified and processed, the partial answers can be applied towards the solution of the general problem.

If the GCS/SF problem is expressed using the Spatial Constraint (SC) graph, strongly constrained clusters of geometric entities can be recognized in the cycles of the SC graph (discussed ahead).

Upon identification of the GCS/SF subproblems, their Grobner Bases can be used in the calculation of a general Grobner Basis instead of the original equations of the subproblem. As a natural step further, *Incremental Instancing* presents the advantage of actually eliminating a degree of freedom from the variable set, therefore contributing to lower the computational expenses of the solution. The intuitive meaning of this action is obviously the fact that, in local instances of the GCS/SF problem, entities may be locked into definite positions, therefore forming clusters which behave as new, unique entities in later stages of the problem. The incremental instancing technique, discussed in chapters 5 and 6 indeed showed the advantages just mentioned over the other techniques applied.

8.2.3.2 Spatial Constraint Graphs

A partition of the GCS/SF problem is required for the statement of a complete, topologically non-redundant polynomial set from the set of constraints. This basic step has to be taken regardless of the subsequent application of D&C techniques. A partition of the GCS/SF problem into subproblems is also needed if D&C techniques are used to solve it. If the GCS/SF is modeled with the help of the SC graph, identification of such subproblems maps to the partition of the SC graph into a *set of basic cycles*. Chapter 5 discussed two issues: (i) counting and identifying sets of basic cycles in the SC graph; and (ii) choosing a convenient set of basic cycles which allows partial, local solutions for the GCS/SF problem. The requirements of the GCS/SF problem demand the partition of the SC graph into a set of short basic cycles. This research has proposed an algorithm that limits the size of the cycles by extracting them with the help of a low-height, large-branching spanning tree for the SC graph. By applying the D&C and Incremental Instancing techniques to problems with large number of entities and complex SC graphs, significant improvements were achieved.

8.3 Recommendations for Future Research

Improvement in the solution techniques for the GCS/SF problem can be achieved by working in the following directions:

- Applications of Divide & Conquer techniques to the solution of this problem are an absolute requirement if real applications are pursued. In chapter 5 a sound theoretical ground for partition of graphs was identified from the literature and applied to the SC graph. Even in the case of numerical solutions for the GCS/SF problem, the use of D&C techniques is promising; in the face of symbolic computation, such as Grobner Basis applications, D&C is frequently the only way to effectively attack a given problem.

- Integer arithmetic, used in the calculation of Grobner Basis, is unstable when applied to floating point problems. A Grobner Basis calculation technique which makes use of symbol re-definition for calculating the coefficients of the polynomials in the Grobner Basis can be used. This solution was not explored in this work because the Grobner Basis calculation routines from packages such as Mathematica, Maple, or Macaulay were used. The possibility of implementing an in-house Grobner Basis Algorithm can be considered for later stages of this research.

- The mapping of characteristics of the solution space onto the physical scene for the GCS/SF problem represents a promising area of research. This process (Oldham coupling example in chapter 7) would allow the formulation of functional conditions on the polynomial ideal corresponding to the problem, and their translation into physical (design) parameters which satisfy those requirements. Notice that the original goal in the GCS/SF problem was to find the solution space corresponding to a basic scenario with constraints. With the proposed methodology, the contrary problem could be undertaken: to re-define the basic scenario and the constraints based on conditions imposed on the solution space.

- The so-called *operational methods* correspond to an intuitive administration of degrees of freedom in the face of sequential application of constraints. These methods are not complete [17, 23, 37] in the sense that they do not guarantee a correct answer in all cases. However, if they are implemented on the theoretical background developed in this investigation, a compromise of speed vs. completeness can be achieved. More importantly, Algebraic Geometry techniques can be used to identify and solve situations in which these methods fail. Algebraic Geometry techniques are most effective for problems whose solution space has low dimension. On the other hand, operational methods have good performance in problems of high dimensionality of the solution space. These relative advantages of the methods could be complemented by each other. For example, some open chain manipulators can be analyzed by an operational approach, while application of Algebraic Geometry

would render no results. On the other hand, cases such as the Bennett mechanism have been reported as out of the reach of operational techniques [23], while it was successfully analyzed by the Grobner Basis method.

Direct application of built-in Grobner Basis routines for floating point manipulation do not produce robust results. Tests run under constant topological (constraint type) conditions and varying geometrical (dimensional) conditions produce execution errors for some of the geometries tried. The natural conclusion in this case is that built-in Grobner Bases routines are not robust when floating point arithmetic is involved. This drawback can be averted if more control is provided on the Grobner Basis calculation. This task would require a strong background both in programming skills and in concepts of algebraic geometry.

REFERENCES

[1] A. Ambler and R. Popplestone. Inferring the Positions of Bodies from Specified Spatial Relationships. *Artificial Intelligence*, 6, 1975.

[2] Jorge Angeles. *Spatial Kinematic Chains*. Springer-Verlag, 1982.

[3] Jorge Angeles. *Rational Kinematics*. Springer-Verlag, 1988.

[4] G. T. Bennet. A New Mechanism. *Engineering*, 1903.

[5] O. Bottema and B. Roth. *Theoretical Kinematics*. North Holland Press, New York, 1979.

[6] B. Buchberger. *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*. PhD thesis, University of Innsbruck, (Austria), 1965.

[7] 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.

[8] E. Celaya and C. Torras. Finding Object Configurations that Satisfy Spatial Relationships . In *European Conference on Artificial Intelligence. Stockholm*, 1990.

[9] Shang-Ching Chou. *Mechanical Geometry Theorem Proving*. Reidel Publishing Co, 1987.

[10] Shang-Ching Chou. Automated reasoning in geometries using the characteristic set method and grobner basis method. In *Proc. International Symposium on Symbolic and Algebraic Computation*, 1990.

[11] J.R. Johnson D. Johnson. *Graph Theory with Engineering Applications.* Ronald Press Company, New York, 1972.

[12] Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall Inc., Englewood Cliffs N.J., 1974.

[13] L. Dobrjanskyj and F. Freudenstein. Some Applications of Graph Theory to the Structural Analysis if Mechanisms. *Journal of Engineering for Industry*, February, 1967.

[14] M.I. Shamos F. Preparata. *Computational Geometry. An Introduction.* Springer Verlag, New York, 1985.

[15] V. M. Faires. *Design of Machine Elements.* MacMillan Company, Toronto, Canada, 1965.

[16] B. Falcidieno and F. Giannini. Automatic Recognition and Representation of Shape-Based Features in a Geometric Modeling System. *Computer Vision, Graphics and Image Processing*, 48:93–123, 1989.

[17] Z. Fu and A. DePennington. Geometric Reasoning Based on Grammar Parsing. *Journal of Mechanical Design*, 116(3), 1994.

[18] M. R. Henderson. Using Syntactic Pattern Recognition to Extract Feature Information from a Solid Geometric Data Base. *Computers in Mechanical Engineering*, pages 61–66, 1983.

[19] J. Herve. Analyse Structurelle des Mechanisms par Groupe des Deplacements. *Mechanism and Machine Theory*, 13, 1978.

[20] Hitochi and Hirochika. Reasoning of geometric concepts based on algebraic constraint-directed method. In *International Joint Conference on Artificial Intelligence*, 1991.

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

[22] 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.

[23] G. Kramer. *Solving Geometric Constraint Systems.* MIT Press, 1992.

[24] L. Kyprianou. Shape Classification in Computer-Aided-Design. Ph.D. thesis, University of Cambridge, 1980.

[25] W Ledermann. *Introduction to the Theory of Finite Groups.* Oliver and Boyd, 1953.

[26] W Ledermann. *Introduction to Group Theory.* Barnes and Noble, 1973.

[27] K. Lee and G. Andrews. Inference of the Positions of Components in an Assembly:Part 2. *Computer Aided Design.*

[28] D.N. Rocheleau and K. Lee. System for Interactive Assembly Modelling . *Computer Aided Design*, 19(2), 1987.

[29] O. Ruiz and P. Ferreira. Algebraic geometry and group theory in geometric constraint satisfaction. In *International Symposium on Symbolic and Algebraic Computation*, St Catherine's College, University of Oxford, UK, July 1994.

[30] 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.

[31] O. Ruiz, R. Marin, and P. Ferreira. A geometric reasoning server with applications to geometric constraint satisfaction and reconfigurable feature extraction. In *3rd Luso-German Workshop on Graphics and Modeling in Science & Technology*, Coimbra, Portugal, June 1994.

[32] ANSI Standard. Dimensioning and Tolerancing. ANSI Standard Y14.5M-1982, 1983.

[33] M. Swamy and K. Thulasiraman. *Graphs, Networks and Algorithms*. John Wiley & Sons, 1974.

[34] F. Thomas and C. Torras. A Group Theoretic Approach to the Computation of Symbolic Part Relations. *IEEE Journal of Robotics and Automation*, 4, 1988.

[35] F. Thomas and C. Torras. Inferring Feasible Assemblies from Spatial Constraints. Technical Report IC-DT-1989.03, Institute of Cybernetics, Inst. Polytecnic of Catalonia, 1989.

[36] Federico Thomas. Graphs of Kinematic Constraints. In L. Homem de Mello and S. Lee, editors, *Computer Aided Mechanical Assembly Planning*, pages 81–109. Kluwer Academic Publishers, 1991.

[37] J. Turner, S. Subramaniam, and S. Gupta. Constraint Representation and Reduction in Assembly Modeling and Analysis. *IEEE Journal of Robotics and Automation*, 8, 1992.

[38] J. Moore V. Chachra, P. Ghare. *Applications of Graph Theory Algorithms*. Elsevier North Holland, New York, 1979.

VITA

Oscar E. Ruiz S. received a B.S. in Mechanical Engineering and a B.S. in Computer Science from Universidad de los Andes, Santa Fé de Bogotá, Colombia, in 1983 and 1987, respectively. From 1984 to 1988 he worked with the Department of Mechanical Engineering in Universidad de los Andes as lecturer and researcher in computer graphics for engineering and computer-aided design and manufacturing. In 1987 he joined an interdisciplinary research group in the Computer Science Department of Universidad de los Andes as coordinator of teaching on informatics for non-computer science disciplines, and as researcher in applied computational geometry.

In 1988 he joined the Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, as graduate student and research assistant. He obtained an M.S. degree in mechanical engineering under the direction of Professors Richard DeVor and Shiv Kapoor in 1991. His thesis project on simulation of interrupted metal cutting was funded by General Motors Company. In 1991 Oscar joined the research group of Professor Placid Ferreira for his Ph.D. work on geometric reasoning for design and manufacturing. He held a summer position with Ford Motor Company in 1993 as a consultant on the development of computer-aided design tools for the Stamping Engineering Division. His research interests include computer-aided design and manufacturing, simulation and visualization of manufacturing processes and computer graphics for technical applications. For his Ph.D. research, he collaborated with faculty at the University of Massachusetts at Amherst, McGill University in Montreal, Ecole Central in Paris, and Universidad Polytecnica de Catalunya in Barcelona. He was invited to deliver seminars about this investigations at several of these institutions.