# Basics on Geometric Constraint Solving

Robert Joan-Arinyo *

**Abstract**

We survey the current state of the art in geometric constraint solving. Both 2D and 3D constraint solving is considered, and different approaches are characterized.

**Keywords**: Geometric constraints, constraint solving, parametric design.

## 1 Introduction and Scope

2D geometric constraint solving is arguably a core technology of computer-aided design (CAD) and, by extension, of managing product design data. Since the introduction of parametric design by Pro/Engineer in the 1980s, every major CAD system has adopted geometric constraint solving into its design interface. Most prominently, 2D constraint solving has become an integral component of sketchers on which most systems base feature design.

Beyond applications in CAD and, by extension, in manufacturing, geometric constraint solving is also applicable in virtual reality and is closely related in a technical sense to geometric theorem proving. For solution techniques, geometric constraint solving also borrows heavily from symbolic algebraic computation and matroid theory.

In this paper, we review basic techniques that are widely available for solving 2D and 3D geometric constraint problems. We focus primarily on the basics of 2D solving and touch lightly on spatial constraint solving and the various ways in which geometric constraint solvers can be extended with relations, external variables, and parameter value enclosures. These and other extensions and problem variants have been published in the literature. They are recommended to the interested reader as follow-on material for study.

## 2 The Geometric Constraint Solving Problem

A geometric constraint problem can be characterized by means of a tuple $(E, O, X, C)$ where $E$ is the geometric space constituting a reference framework into which the problem is embedded. $E$ is usually Euclidean, $O$ is the set of specific geometric objects which define the problem. They are chosen from a fixed repertoire including points, lines, circles and the like, and $C$ is the set of geometric constraints. They are relationships between geometric elements chosen from a predefined set, e.g., distance, angle, tangency, etc.

The geometric constraint solving problem can now be stated as follows: Given a set $O$ with $n$ geometric elements and a set $C$ with $m$ geometric constraints defined on them

1. Is there a placement of the $n$ geometric elements such that the $m$ constraints are fulfilled? If the answer is positive,

*Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Universitat Politècnica de Catalunya robert@lsi.upc.edu

2. given an assignment of values to the $m$ constraints tags, is there an actual construction of the $n$ geometric elements satisfying the constraints?

When dealing with geometric constraint solving, the first issue that needs to be settled is the dimension of the embedding space $E$. In 2D Euclidean space, $E = \mathcal{R}^2$, a number of techniques have been developed that successfully solve the geometric constraint solving problem. For an in-depth review see Jermann, [29]. However, there remain open questions such as characterizing the competence (also called domain) of the known techniques.

Spatial constraint solving, where $E = \mathcal{R}^3$, include problems in fields like molecular modeling, robotics, and terrain modeling. Here, both a good conceptualization and an effective solving methodology for the geometric constraint problem has proved to be difficult. Pioneering work has been reported by Hoffmann and Vermeer, [27] and by Durand, [13].

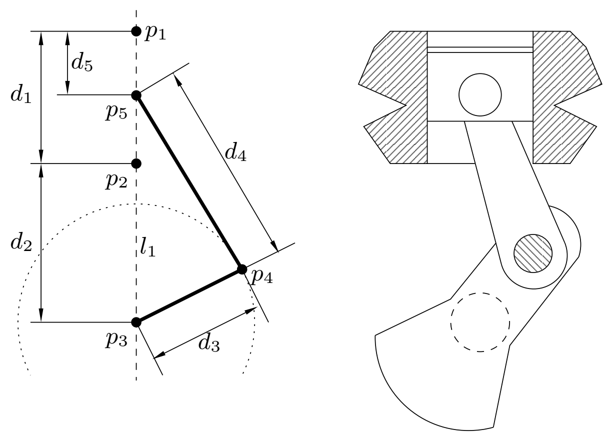


Figure 1: Piston, crankshaft and connecting rod mechanism.

Figure 1 depicts a piston-crankshaft mechanism, [12], a geometric constraint solving problem in 2D. The left side shows the geometric problem, the right side shows the actual mechanism so abstracted. The mechanism transforms the translational motion of point $p_5$ along the straight line $l_1$ into a rotational motion of point $p_4$, on a circular path with center $p_3$ and radius $d_3$. The piston-crankshaft mechanism can be abstracted as a geometric constraint solving problem comprising five points $p_i$, $1 \leq i \leq 5$, and a straight line $l_1$. The set of constraints is given in Figure 2 and includes point-point distances, *dpp*(), and coincidences, *on*().

## 2.1 Problem Categorization

The CAD/CAM community focuses on the design and manufacture of rigid objects, that is, objects that are fully determined up to a global coordinate system. Similarly, we seek solutions to a constraint problem that are determined up to a global coordinate system, that is, where solutions are congruent under the rigid-body transformations of translation and rotation. We call a configuration of geometric objects in Euclidean space *rigid* when all objects are fixed with respect to each other up to translation and rotation.

| | | | |
|---|---|---|---|
| 1. | $\text{dpp}(p_1, p_2) = d_1$ | 6. | $\text{on}(p_1, l_1)$ |
| 2. | $\text{dpp}(p_2, p_3) = d_2$ | 7. | $\text{on}(p_2, l_1)$ |
| 3. | $\text{dpp}(p_3, p_4) = d_3$ | 8. | $\text{on}(p_3, l_1)$ |
| 4. | $\text{dpp}(p_4, p_5) = d_4$ | 9. | $\text{on}(p_5, l_1)$ |
| 5. | $\text{dpp}(p_1, p_5) = d_5$ | | |

Figure 2: A set of geometric constraints for the piston-crankshaft mechanism.

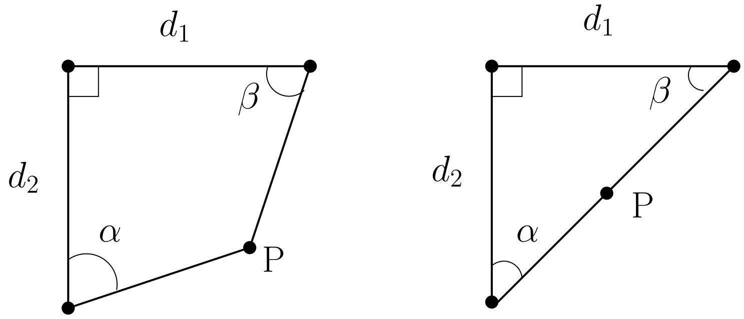


Figure 3: Left: General configuration. Right: Degenerate configuration for $\alpha + \beta = 90°$. Fudos and Hoffmann, [15].

An intuitive way to introduce rigidity comes from considering the number of solutions that a geometric constraint problem has. There are three categories: A problem is *structurally under constrained* if there are infinitely many solutions that are not congruent under rigid transformation, *structurally well-constrained*, if there are finitely many solutions modulo rigid transformation, and *structurally over constrained* if the deletion of one or more constraints results in a well-constrained problem. A constraint problem naturally corresponds to a set of (usually nonlinear) algebraic equations.

Defined in this way, the concept of rigidity appears to be simple but it is not quite in accord with the intuition about rigidity. The categories so defined only refer to the problem's structure and do not account for other issues such as inconsistencies that could originate from specific values assigned to the constraints. Clearly a problem that is structurally well-constrained could actually be underconstrained for specific values of the constraints. For example, consider the *structurally* well constrained problem given in Figure 3, see Fudos and Hoffmann, [15]. Point $P$ is properly placed whenever $\alpha + \beta \neq 90°$ and the problem is well-constrained. But if $\alpha + \beta = 90°$, then the placement for point $P$ is undetermined and, therefore, the problem is no longer well constrained.

Different formal definitions of rigidity have been explored in the literature. See, for example, the work by Henneberg, [19], and Laman, [40], or the more recent works by Graver *et al.* [17], Fudos and Hoffmann, [15], Hoffmann *et al.*, [24], and Whitley, [64].

# 3 Major Approaches

Geometric constraint solving methods can be roughly classified as graph-based, logic-based, or algebraic. For 2D solvers, the graph-based approach has become dominant in CAD. A problem closely related to geometric constraint solving is Automated Theorem Proving.

## 3.1 Graph-Based Approach

In the graph-based approach, the constraint problem is translated into a graph (or hyper graph) whose vertices represent the geometric elements and whose edges the constraints upon them . The solver analyzes the graph and formulates a solution strategy whereby subproblems are isolated and their solutions suitably combined. A subsequent phase then solves all subproblems and combines them. The advantage of this type of solver is that the subproblems often are very small and fall into a few simple categories. The disadvantage is that the graph analysis of a fully competent solver is rather complicated. The graph-based approach can be further subdivided into constructive, degree of freedom analysis, and propagation.

### 3.1.1 Constructive Approach

The constructive approach generates the solution to a geometric constraint problem as a symbolic sequence of basic construction steps. Each step is a rule taken form a predefined set of operations that position a subset of the geometric elements. For example, the operations may restrict to ruler-and-compass constructions. Clearly, this approach preserves the geometric sense of each operation involved in the solution. Note that the sequence of construction steps allows to compactly represent a possibly exponential number of solution instances. However, the constructive approach cannot solve problems with symbolic constraints or external variables.

Depending on the technique used to analyze the problem, two different categories of constructive approaches can be distinguished: *top-down* and *bottom-up*.

The top-down technique recursively splits the problem until it has isolated simpler, basic problems whose solutions are known. In this category, Todd, [60], defines the *r-tree* concept and derives a geometric constraint solving algorithm. Owen, [51], describes a more general method based on the recursive decomposition of the constraints graph into triconnected components. Inspired by Owen's work, Fudos reported a new decomposition method in [15]. Efficient algorithms with a running time $O(n^2)$, where $n$ is the number of geometric elements, are known for the methods of Owen, [51], and Fudos, [15].

In the bottom-up approach, the solution is built by suitably combining recursively solutions to sub-problems already computed, starting from the constraints in the given set, considering each constraint as a single element.

Constraints may be represented implicitly as a collection of sets of geometric elements where the elements of each set are placed with respect to a local framework. Sets are merged; e.g., by application of rewriting rules until all the geometric elements are included in just one set. The advantage of this representation is that the sets of constraints capture the relationships between geometric elements compactly. Fudos *et al.*, in the method described in [15], use one type of sets of constraints, called *cluster*, and one generic rule that merges three clusters which pairwise share two elements.

Lee *et al.*, [43], describe a constructive method that associates with each vertex in the graph a status which can be defined, half defined or not defined. Inference rules are used to modify the status of the vertices.

Efficient algorithms with a running time $O(n^2)$, where $n$ is the number of geometric elements, are known for the methods listed above. However, the constructive approach is not complete, therefore assessing the competence of solvers in this category is an important issue. Verroust, [63], partially characterizes the set of relevant problems solved by the solver described. Joan-Arinyo *et al.*, [32], describe a formalization that unifies the methods reported by Fudos, [15], and Owen, [51]. In [33], Joan-Arinyo *et al.* show that the sets of problems solved by Fudos' and Owen's approaches are the same.

In [23] Hoffmann and Joan-Arinyo describe a technique that extends constructive methods with the capability of managing functional relationships between geometric and externals variables. Essentially, the technique combines two methods: one is a constructive constraint solving method, and the other is a systems of equations analysis method. Joan-Arinyo and Soto-Riera, [31], further improved the technique and formalized it as a rewriting system.

Constructive methods work well in 2-space. Several attempts to extend them to 3-space have been reported. See, for example, Brüderlin [6], Verroust [62], and Hoffmann and Vermeer [27].

### 3.1.2 Degrees of Freedom Analysis

Degrees of Freedom Analysis assigns degrees of freedom to the geometric elements by labeling the vertices of the graph of the problem. Each edge of the graph is labeled with the number of degrees of freedom canceled by the associated constraint. Then the method solves the problem by analyzing the resulting labeled graph.

Kramer, [38], developed a method to solve specific problems from the kinematics of mechanisms. The method applies techniques borrowed from the process planning field to yield a symbolic solution. Since the set of rules used to generate the plan preserves geometric sense, the entire method also preserves it. Kramer, [38], proves that his method is correct by showing that the set of rules together with the labeled graph is a canonical rewriting system. The method runs in time $O(nm)$, where $n$ is the number of geometric elements and $m$ the number of constraints in the problem. Since $m$ is typically $O(n)$, the method has the same complexity as the constructive approach. Bhansali *et al.*, [3], describe a method that generates automatically segments of the symbolic solution in Kramer's approach.

Salomons *et al.*, [52], represent objects and constraints as a graph and apply geometric and equational reasoning following the lines given by Kramer's method.

In [28], Hsu reports a method with two phases. First, a symbolic solution is generated. Then, the actual construction is carried out. The method applies geometric reasoning and, if this fails, numerical computation.

Latham *et al.*, [42], decompose the labeled graph in minimal connected components called *balanced sets*. If a balanced set corresponds to one of the predefined specific geometric constructions, then it can be solved. Otherwise the underlying equations are solved numerically. The method also deals with symbolic constraints and identifies over- and under constrained problems. Assigning priorities to the constraints allows them to solve over constrained problems. A proof of correctness is also given.

Hoffmann *et al.*, [25], have developed a flow-based method for decomposing graphs of geometric constraint problems. The method generically iterates to obtain a decomposition of the underlying algebraic system into small subsystems called *minimal dense* subgraphs. The method fully generalizes degree-of-freedom calculations, the approaches based on matching specific subgraphs patterns, as well as the prior flow-based approaches. However, the decomposition rendered does not necessarily have geometric sense since minimal dense subgraphs can be of arbitrary complexity far exceeding problems that yield to classical geometric construction.

#### 3.1.3 Propagation Approach

Propagation methods represent the set of algebraic equations with a symmetric graph whose vertices are variables and equations and whose edges are labeled with the occurrences of the variables in the equations.

Propagation methods try to orient the edges in the graph in such a way that each equation vertex is a sink for all the edges incident on it except one. If the process succeeds, then there is a general incremental solution. That is, the system of equation can be transformed into a triangular system and solved using back substitution.

Among the techniques to orient a graph we find in the literature degrees of freedom propagation and propagation of known values, [14, 53, 61]. Propagation methods do not guarantee finding a solution whenever one exists. They fail when the orientation algorithm finds a loop. Propagation methods can combined with numerical methods for equation solving to ameliorate circularity, [5, 39, 56, 59]. Veltkamp and Arbab, [61], apply other techniques to break loops created while orienting the graph.

Leler, [44], describes propagation methods in depth and proposes *augmented rewriting terms*, a tool which consists of a classical rewriting system along with an association of atomic-value and object-type. This tool has had success in solving certain systems of nonlinear equations.

In [4], Borning *et al.*, describe an local propagation algorithm that can deal with inequalities.

### 3.2 Logic-Based Approach

In the logic-based approach, the problem is translated into a set of assertions and axioms characterizing the constraints and the geometric objects. By employing reasoning steps, the assertions are transformed in ways that expose solution steps in a stereotypical way and special solvers then compute coordinate assignments.

Aldefeld, [1], Brüderlin, [7], Sohrt, [55] and, Yamaguchi *et al.*, [66], use first order logic to derive geometric information applying a set of axioms from Hilbert's geometry. Essentially these methods yield geometric loci at which the elements must be.

Sunde, [58], and Verroust, [63], consider two different types of sets of constraints: sets of points placed with respect to a local framework, and sets of straight line segments whose directions are fixed with respect to a local framework. The reasoning is basically performed by means of a rewriting system on the sets of constraints. The problem is solved when all the geometric elements belong to a unique set. Joan-Arinyo and Soto-Riera, [30], extended these sets of constraints with a third type consisting of sets containing one point and one straight line such that the perpendicular point-line distance is fixed.

## 3.3 Algebraic Methods

In the algebraic approach, the constraint problem is translated directly into a set of nonlinear equations and is solved using any of the available methods for solving nonlinear equations. The main advantages of algebraic solvers are their generality, dimension independence and the ability to deal with symbolic constraints naturally.

In principle, an algebraic solver can be fully competent. However, algebraic solvers may have low efficiency or may have difficulty constructing solutions reliably. When used to pre-process and study specific constraint systems, however, algebraic techniques can be extremely useful and very practical.

As a result of mapping the geometric domain problem into an equational one, the geometric sense of the solutions rendered is lost. Moreover, well constrained problems are mapped to under constrained systems of equations because constraints fix the placement for each geometric element with respect each other only modulo translation and rotation. Therefore a set of additional equations must be joined to cancel these remaining degrees of freedom.

Algebraic methods can be further classified according to the specific technique used to solve the system of equations, namely into numerical, symbolic, and analysis of systems of equations.

### 3.3.1 Numerical Methods

Numerical methods provide powerful tools to solve iteratively large systems of equations. In general, a good approximation of the intended solution should be supplied to guarantee convergence. This means that if, as it is customary, the starting point is taken from the sketch defined by the user, then the sketch should be close to the intended solution. The numerical methods may offer little control over the solution in which the user is interested. To achieve robustness, numerical iterative methods must be carefully designed and implemented.

Borning, [5], Hillary and Braid, [21], and Sutherland, [59] use a *relaxation* method. This method is an alternative to the propagation method. Basically, the method perturbs the values assigned to the variables and minimizes some measure of the global error. In general, convergence to a solution is slow.

The method most widely used is the well-known *Newton-Raphson*, [34] iteration. It is used in the solvers described in [20, 45, 46, 50]. Newton-Raphson is a local method and converges much faster than relaxation. The method does not apply to consistently over constrained systems of equations unless special provisions are made such as combining it with least-squares techniques.

*Homotopy* or *continuation*, [2], is a family of methods with a growing popularity. These methods are global and guarantee convergence. Moreover, they are exhaustive and allow to determine all solutions of a constraint problem. However, their efficiency is worse than that of Newton-Raphson. Lamure and Michelucci, [41], and Durand, [13], apply this method to geometric constraint solving.

Other, less conventional methods have also been proposed. For example, in [18], Hel-Or *et al.*, introduced the *relaxed parametric design* method where the constraints are soft, that is, they do not have to be met exactly, the problem is modeled as a static stochastic process, and the resulting system of probabilistic equations is solved using the Kalman filter familiar from control theory. The Kalman filter was developed to efficiently compute linear estimators and when applied to nonlinear systems, it does not necessarily finds a solution even if one exists.

Kin *et al.*, [35], reported on a numerical method based on extended Boltzmann machines which are a sort of neural network whose goal is to minimize a given polynomial that measures the energy of the system.

#### 3.3.2 Symbolic Methods

Symbolic algebraic methods compute a Gröbner basis for the given system of equations. Algorithms to compute these bases include those by Buchberger [9], and by Wu-Ritt [10, 65]. These methods, essentially, transform the system of polynomial equations into a triangular system whose solutions are those of the given system. In effect, triangularization reduces solving a simultaneous, nonlinear system to univariate root finding. Forward or a backward substitution must be used.

Buchanan *et al.*, [8], describe a solver built on top of the Buchberger's algorithm. In [36], Kondo reports a symbolic algebraic method. In [37], Kondo improves that work by generating a polynomial that summarizes the changes undergone by the system of equations.

#### 3.3.3 Analysis of Systems of Equations

Methods based on the analysis of systems of equations determine whether a system is under-, well- or over-constrained from the system structure. These methods can be extended to decompose systems of equations into a set of minimal graphs which can be solved independently, [49, 57]. They can be used as a pre-processing phase for any other method, reducing the number of variables and equations that must be solved simultaneously.

Serrano, [54], applies analysis of systems of equations to select from a set of candidate constraints a well constrained, solvable subsets of equations.

### 3.4 Theorem Proving

Solving a geometric constraint problem can be seen as automatically proving a geometric theorem. However, automatic geometric theorem proving requires more general techniques and, therefore, methods which are much more complex than those required by geometric constraint solving.

Wu Wen Tsün pioneered the Wu-Ritt method, an algebraic-based geometric constraint solving method. In [65], he uses it to prove geometric theorems. The method automatically finds necessary conditions to obtain non-degenerated solutions.

In [10], Chou applies Wu's method to prove novel geometric theorems. Chou *et al.*, in [11], report on a work in automatic geometric theorem proving which allows to interpret, from a geometric point of view, the proof generated by computation.

## 4 Spatial Constraint Solving

In 2D sketching applications, a major application area of constraint solving, graph-based solvers have become dominant. The underlying reason is that for the constructs common in 2D sketching a small set of subgraphs suffices, and that the associated algebraic solution problems are rather simple. In constrast, spatial constraint solving does not appear to have a simple and small core that suffices for most practical applications.

When approached using graph decomposition, the problem of spatial constraint solving is that simple subgraphs of, even up to 6 vertices, are numerous and many of them correspond to associated algebraic problems that are rather difficult. Note that a subgraph with 6 vertices is the smallest simultaneous spatial constraint problem involving only points and planes.

There is also the problem that no consensus has arisen of the characteristic spatial constraint problems of relevance to, say, CAD, so that there is little guidance on how to select a subset from the subgraph patterns to arrive at a compact solver that is widely applicable. In this section we review some of these issues in the context of a graph decomposition solver architecture. We begin with the problem of solving the equations associated with a selection of spatial constraint problems.

### 4.1 Sequential Construction Problems

The simplest 3-space constraint problems require placing a single geometric element (point, plane or line) with respect to a set of geometric elements whose position and orientation are known. We call such problems *sequential*, since the elements are placed one-by-one sequentially.

Many, but not all, sequential problems are easy to solve. For example, placing a point with respect to three known points requires the intersection of three spheres. Elementary algebraic manipulation reduces this task to solving a univariate quadratic equation. On the other hand, a difficult sequential problem is placing a line such that it is at prescribed distance from four known points in 3-space. Geometrically, this is equivalent to finding common tangents to four given spheres.

A lower bound on the number of tangents to four spheres was established by Macdonald *et al.*, [47], who exhibited four unit spheres that have 12 distinct, common tangents. That this bound is sharp was proved by Hoffmann and Yuan in [22] by deriving a nonlinear system of equations whose solutions give all common tangents. It was shown that the geometric degree of the system is 12, thus establishing the missing upper bound. Note that solving this system is not trivial.

### 4.2 Algebraic and Geometric Solutions

In the algebraic approach to solving specific constraint problems, one formulates a system of algebraic equations. The system is then simplified using algebraic manipulation and geometric reasoning, and if the resulting system is simple enough, its solutions can be computed with high reliability and accuracy. For spatial constraint problems, it is not necessarily easy to find such simple systems, and so computing solutions may require sophisticated algorithms, e.g., [13]. As we have seen, sequential problems may already require solving algebraic systems of degree 12.

A geometric aspproach to finding the solutions of a nonlinear equation system is the *locus method*, [16]. Instead of relying only on root finding techniques, a geometric idea is introduced:

> Drop one constraint from the problem, say a dimensional constraint $c$, resulting in an underconstrained problem. Evaluate the underconstrained configuration and measure the actual value of the dimension. Different configurations lead to different values, resulting in a curve that can be traced. This curve is the *locus* of $c$. Intersect the locus of $c$ with the nominal constraint value, usually a straight line. The resulting intersection configurations are solutions to the original system.

The method can be extended to computing the locus of more than one cut constraint, resulting in geometric manifolds of higher dimensions. Cutting two constraints we would obtain a surface that would be intersected with two planes, cutting three constraints a spatial manifold is obtained, and so on. We will illustrate the idea for octahedral problems.

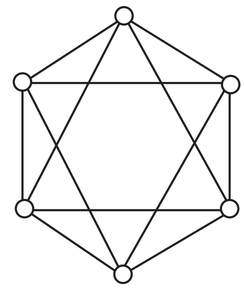

Figure 4: Octahedral graph.

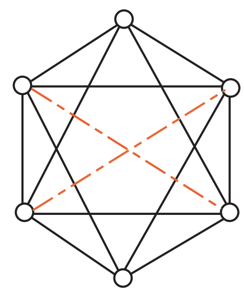

Figure 5: Octahedral graph. In red, dependent distances.

### 4.3 Octahedral Problems

Consider a constraint graph with six vertices and with edges arranged as shown in Figure 4. We call such a graph *octahedral* on account of the fact that the topology is that of the vertices and edges of a regular octahedron. The vertices represent points or planes in 3-space, and the edges therefore a distance between two points, a distance between a point and a plane, and/or an angle between two planes. There are seven major configurations according to the number of planes in the problem. Two of them, namely configurations with 5 or 6 planes, are structurally under determined. The other five configurations can be solved algebraically; [26].

An elegant approach to formulating the equation system is due to Michelucci, [48][1] who employs the Cayley-Menger determinant as follows. The determinant expresses the distances between the five points:

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12} & d_{13} & d_{14} & d_{15} \\ 1 & d_{12} & 0 & d_{23} & d_{24} & d_{25} \\ 1 & d_{13} & d_{23} & 0 & d_{34} & d_{35} \\ 1 & d_{14} & d_{24} & d_{34} & 0 & d_{45} \\ 1 & d_{15} & d_{25} & d_{35} & d_{45} & 0 \end{vmatrix} = 0$$

where the $d_{ik} = \|p_i - p_k\|^2$ is the squared distance between the points $p_i$ and $p_k$. Choosing two sets of five points in the configuration, we can express the squared distances indicated by a red, dashed line in Figure 5 as function of the given distances. This results in two bivariate equations of degree 4 each in the two unknown distances, indicating at most 16 distinct solutions. That the bound of 16 solutions is sharp was shown earlier by Hoffmann and Vermeer in [26]. Suitable extensions of the Cayley-Menger determinant can be used for the remaining cases comprising both points and planes.

A solution of the two quartic equations can be obtained by the locus method, for example. Each

[1]The URL is no longer valid.

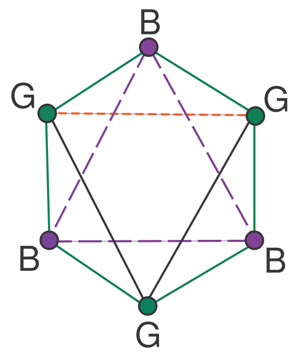


Figure 6: Octahedral problem. The locus method.

quartic equation taken separately yields a plane curve. Tracing these curves can then give initial values for an iterative numerical method that isolates the up to 16 roots of the system.

The locus method can also be used to plot directly the value of a cut constraint, and we explain the approach assuming only points in the configuration. Consider Figure 6. Select the three blue vertices, labeled B, and cut the red, dotted-line constraint (between two green vertices labeled G). The three blue vertices B can be positioned using the blue constraints only, indicated by dashed lines, in the $xy$-plane for example.

Each of the green vertices, labeled G in the figure, together with the adjacent blue vertices, labeled B, can be thought of as a rigid triangle that pivots about a blue, dashed edge, and so each green vertex moves on a circle whose plane is perpendicular to the $xy$-plane. Select one green vertex and parameterize its position on the circle on which it moves, say by $\theta$. For a given $\theta$ value, we can now compute the position of the remaining two green vertices using the black constraints, indicated by solid lines between the G vertices, intersecting a circle and a sphere. Once these vertices have been placed, we can compute the distance $d$ between the two vertices adjacent to the cut, red constraint, so plotting a curve in the $\theta d$-plane. Intersecting the curve with the line $d = d_0$, where $d_0$ is the distance stipulated by the red constraint, we obtain all solutions to the original problem.

### 4.4 Line Configurations

In the discussion of sequential problems before, we have seen that sequential line problems in 3-space may yield algebraic equation systems of considerable complexity. In addition, there is a large number of individual problems involving only a small number of geometric elements. We illustrate the combinatorial explosion of cases involving constraint graphs with 6 or fewer vertices representing points, lines and planes.

Note that between two planes, between to points, and between a point and a plane we can have at most one constraint, namely of angle, of distance, and of distance, respectively. Hoffmann and Vermeer show in [26] that the octahedron problem is the simplest, nonsequential problem involving only points and planes. As shown in that paper, there are exactly seven distinct major[2] configurations, two of them underconstrained.

We can have at most a single constraint between a point and a line (distance), and a single constraint between a plane and a line (angle). However, between two lines we can have up to two constraints (distance and angle). As Gao *et al.* show in [16], there are two distinct nonsequential constraint problems with four lines, shown in Figure 7.

[2]Some of the major configurations have several sub configurations. For instance, in a problem with two planes, the planes could have an angle constraint between them, or no constraint them, leading to two sub configurations for this case.

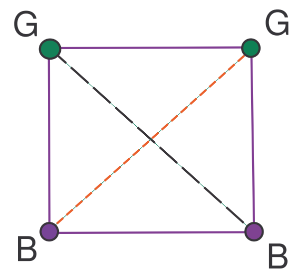


Figure 7: The four lines problem.

The blue, solid lines represent two constraints, of angle and distance, and the black, dashed line represents a distance constraint. The two configurations differ in the red, dotted line constraint, a distance constraint in one and an angle constraint in the other configuration.

In [16], Gao *et al.* also establish that there are 17 distinct configurations with 5 geometric elements, including lines, but more than 680 configurations with 6 geometric elements. These numbers show that a solver, based on decomposing a spatial constraint problem into a (recursive) set of small subproblems must have a very large repertoire of subproblem patterns, even when allowing only up to six geometric elements. Moreover, the algebraic structure of the many configurations involving lines remain unexplored.

## Acknowledgements

I wish to acknowledge my collaborators over the years, especially M. Freixas, I. Fudos, C. M. Hoffmann, A. Soto, M. Tarrès, Sebastià Vila. Much of the material reviewed has been to their credit. R. Joan-Arinyo has been supported by The Ministerio de Educación y Ciencia and by FEDER under grant TIN2007-67474-C03-01.

## References

[1] B. Aldefeld. Variation of geometric based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, April 1988.

[2] Eugene L. Allgower and Kurt Georg. Continuation and path following. *Acta Numerica*, 7:1–64, 1993.

[3] S. Bhansali, G.A. Kramer, and T.J. Hoar. A principled approach towards symbolic geometric constraint satisfaction. *Journal of Artificial Intelligence Research*, (4):419–443, 1996.

[4] A. Borning, R. Anderson, and B. Freeman-Benson. Indigo: A local propagation algorithm for inequality constraints. In *UIST'96*, pages 129–136, Seattle, Washington, USA, November 6-8 1996. ACM.

[5] A.H. Borning. The programming language aspects of ThingLab, a constrained oriented simulation laboratory. *ACM Trans. on Prog. Lang. and Systems*, 3(4):353–387, October 1981.

[6] B.D. Brüderlin. Constructing three-dimensional geometric objects defined by constraints. In *Workshop on Interactive 3D Graphics*, pages 111–129. ACM, October 23–24 1986.

[7] B.D. Brüderlin. Using geometric rewrite rules for solving geometric problems symbolically. In *Theoretical Computer Science 116*, pages 291–303. Elsevier Science Publishers B.V., 1993.

[8] S.A. Buchanan and A. de Penington. Constraint definition system: a computer-algebra based approach to solving geometric-constraint problems. *Computer-Aided Design*, 25(12):741–750, December 1993.

[9] B. Buchberger. *Multidimensional Systems Theory*, chapter Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, pages 184–232. D. Reidel Publishing Theory, 1985.

[10] S.-C. Chou. An introduction to Wu's method for mechanical theorem proving in geometry. *Journal of Automated Reasoning*, 4:237–267, 1988.

[11] S.-C. Chou, X.-S. Gao, and J.-Z. Zhang. Automated generation of readable proofs with geometric invariants: Multiple and shortest proof generation. *Journal of Automated Reasoning*, 7:325–347, 1996.

[12] H.M. Cundy and A.P. Rollet. *Mathematical Models*. Oxford University Press, 1961. Second edition.

[13] C. Durand. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Computer Science, Purdue University, December 1998.

[14] B. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, 1990.

[15] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, April 1997.

[16] X.-S. Gao, C. Hoffmann, and W.-Q. Yang. Solving spatial basic geometric constraint configurations with locus intersection. In *Solid Modeling SM'02*, Saarbrücken, Germany, 2002.

[17] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*. American Mathematical Society, 1993.

[18] Y. Hel-Or, A. Rapoport, and M.Werman. Relaxed parametric design with probabilistic constraints. In J. Rossignac, J. Turner, and G. Allen, editors, *Second Symposium on Solid Modeling and Applications*, pages 261–270, Montreal, Canada, May 19-21 1993. ACM Press.

[19] L. Henneberg. *Die Graphische Statik de Starren Systeme*. Leipzig, 1911.

[20] A. Heydon and G. Nelson. The Juno-2 constraint-based drawing editor. Research Report 131a, Digital Systems Research Center, December 1994.

[21] R.C. Hillyard and I.C. Braid. Characterizing non-ideal shapes in terms of dimensions and tolerances. In *ACM Computer Graphics*, pages 234–238, 1978.

[22] C. Hoffmann and B. Yuan. There are 12 common tangents to four spheres. http://www.cs.purdue.edu /homes/cmh/distribution/ SphereTangents.htm, 2000.

[23] C.M. Hoffmann and R. Joan-Arinyo. Symbolic constraints in constructive geometric constraint solving. *Journal of Symbolic Computation*, 23:287–300, 1997.

[24] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming*, pages 463–477, Schloss Hagenberg, Austria, October 29 - November 1 1977.

[25] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Decompostion Plans for Geometric Constraint Systems, Part I: Performance Measurements for CAD. *Journal of Symbolic Computation*, 31:367–408, 2001.

[26] C.M. Hoffmann and P.J. Vermeer. Geometric constraint solving in $\mathrm{R}^2$ and $\mathrm{R}^3$. In D.-Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, pages 266–298. World Scientific Publishing, 1995.

[27] C.M. Hoffmann and P.J. Vermeer. A spatial constraint problem. In J.P. Merlet and B. Ravani, editors, *Computational Kinematics'95*, pages 83–92. Kluwer Academic Publ., 1995.

[28] C.-Y. Hsu and B.D. Brüderlin. A hybrid constraint solver using exact and iterative geometric constructions. In D. Roller and P. Brunet, editors, *CAD Systems Development: Tools and Methods*, pages 265–279, Berlin, 1997. Springer-Verlag.

[29] C. Jermann. *Résolution de constraintes géométriques par rigidification récursive et propagation d'intervalles*. PhD thesis, Université de Nice-Sophia Antipolis, 2002. (Written in French).

[30] R. Joan-Arinyo and A. Soto. A correct rule-based geometric constraint solver. *Computer & Graphics*, 21(5):599–609, 1997.

[31] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational geometric constraint solving techniques. *ACM Transactions on Graphics*, 18(1):35–55, January 1999.

[32] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Declarative characterization of a general architecture for constructive geometric constraint solvers. In D. Plemenos, editor, *The Fifth International Conference on Computer Graphics and Artificial Intelligence*, pages 63–76, Limoges, France, 14-15 May 2002. Université de Limoges.

[33] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Revisiting decomposition analysis of geometric constraint graphs. In K. Lee and N. Patrikalaikis, editors, *Seventh Symposium on Solid Modeling and Applications*, pages 105–115, Saarbrücken, Germany, June 19-21 2002. ACM Press.

[34] L. W. Johnson and R. D. Riess. *Numerical analysis*. Addison-Wesley, 1982. Second edition.

[35] N. Kin, Y. Takai, and T.L. Kunii. A connectionist approach to geometrical constraint-solving. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics*. Springer Verlag, 1993.

[36] K. Kondo. PIGMOD : Parametric and interactive geometric modeller for mechanical design. *Computer Aided Design*, 22(10):633–644, December 1990.

[37] K. Kondo. Algebraic method for manipulation of dimensional relationships in geometric models. *Computer Aided Design*, 24(3):141–147, March 1992.

[38] G. Kramer. *Solving Geometric Constraints Systems*. MIT Press, 1992.

[39] G. Kwaiter, V. Gaildrat, and R. Caubet. Interactive constraint system for solid modeling objects. In C. Hoffmann and W. Bronsvoort, editors, *Fourth Symposium on Solid Modeling and Applications*, pages 265–270. ACM SIGGRAPH, May 1997.

[40] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, October 1970.

[41] H. Lamure and D. Michelucci. Solving geometric constraints by homotopy. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 263–269, Salt Lake City, Utah USA, May 17-19 1995. ACM Press.

[42] R.S. Latham and A.E. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28(11):917–928, November 1996.

[43] J.Y. Lee and K. Kim. Geometric reasoning for knowledge-based parametric design using graph representations. *Computer-Aided Design*, 28(10):831–841, 1996.

[44] W. Leler. *Constraint Programming Languages: Their Specification and Generation*. Addison Wesley, 1988.

[45] R. Light and D. Gossard. Modification of geometric models through variational geometry. *Computer Aided Design*, 14:209–214, July 1982.

[46] V.C. Lin, D.C. Gossard, and R.A. Light. Variational geometry in computer-aided design. *ACM Computer Graphics*, 15(3):171–177, August 1981.

[47] I. Macdonald, J. Pach, and T. Theobald. Common tangents to four unit balls. *Discr. Comp. Geometry*, 26:1–17, 2001.

[48] D. Michelucci. Using Cayley-Menger determinants. http://www.emse.fr/~micheluc/MENGER/. approx. from 2000; URL no longer valid.

[49] K. Murota. *Systems Analysis by Graphs and Matroids.* Algorithms and Combinatorics 3. Springer-Verlag, 1987.

[50] G. Nelson. Juno, a constraint-based graphics system. *SIGGRAPH*, pages 235–243, San Francisco, July 22–26 1985.

[51] J.C. Owen. Algebraic solution for geometry from dimensional constraints. In R. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 397–407, Austin, TX, June 5-7 1991. ACM Press.

[52] O.W. Salomons, F. van Slooten, F.J.A.M van Houten, and H.J.J. Kals. Conceptual graphs in constraint based re-design. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 55–64, Salt Lake City, Utah USA, May 17-19 1995. ACM Press.

[53] M. Sannella. The SkyBlue constraint solver. Technical Report 92-07-02, University of Washington, Dep of Computer Science and Engineering, 1993.

[54] D. Serrano. Automatic dimensioning in design for manufacturing. In J. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 379–386, Austin, TX, June 5-7 1991. ACM Press.

[55] W. Sohrt and B.D. Brüderlin. Interaction with constraints in 3D modeling. *International Journal of Computational Geometry & Applications*, 1(4):405–425, 1991.

[56] L. Solano and P. Brunet. Constructive constraint-based model for parametric CAD systems. *Computer Aided Design*, 26(8):614–621, August 1994.

[57] N. Sridhar, R. Agrawal, and G.L. Kinzel. Algorithms for the structural diagnosis and decomposition of sparse, underconstrained design systems. *Computer-Aided Design*, 28(4):237–249, 1996.

[58] G. Sunde. A CAD system with declarative specification of shape. *Eurographics Workshop on Intelligent CAD Systems*, pages 90–105, April 1987.

[59] I. Sutherland. Sketchpad, a man-machine graphical communication system. In *Proc. of the Spring Joint Comp. Conference*, pages 329–345. IFIPS, 1963.

[60] P. Todd. A k-tree generalization that characterizes consistency of dimensioned engineering drawings. *SIAM J. Disc. Math*, 2(2):255–261, 1989.

[61] R.C. Veltkamp and F. Arbab. Geometric constraint propagation with quantum labels. In B. Falcidieno, I. Herman, and C. Pienovi, editors, *Eurographics Workshop on Computer Graphics and Mathematics*, pages 211–228. Springer, 1992.

[62] A. Verroust. *Etude de Problèmes Liés à la Définition, la Visualisation et l'Animation d'Objects Complexes en Informatique Graphique.* PhD thesis, Universite de Paris-Sud, Centre d'Orsay, 1990. (Written in French).

[63] A. Verroust, F. Schonek, and D. Roller. Rule-oriented method for parameterized computer-aided design. *Computer Aided Design*, 24(10):531–540, October 1992.

[64] W. Whiteley. Rigidity and scene analysis. In J.E. Goodman and J. O'Rourke, editors, *Handbook for discrete and computational geometry*, pages 893–916. CRC Press LLC, 1998.

[65] W.-T. Wu. Mechanical theorem proving in geometries. In B. Buchberger and G. E. Collins, editors, *Texts and monographs in symbolic computations.* Springer-Verlag, 1994.

[66] Y. Yamaguchi and F. Kimura. A constraint modeling system for variational geometry. In J.U. Turner M.J. Wozny and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 221–233. Elsevier North Holland, 1990.