Variational Constraints in 3D

Cassiano Durand Christoph M. Hoffmann

Computer Science Department
Purdue University
West Lafayette, IN 47907-1398
USA

Abstract

We discuss how to solve variational constraint problems involving points, lines and planes. We concentrate on small problems that must be solved simultaneously.

1. Introduction

Geometric constraint solving is an integral part of computer aided design, serving the role both of precisely situating geometric elements in relation to each other, as well as dimensioning and constraining their shapes accurately. There is a large literature on the subject, addressing the problem with a variety of approaches. For a recent comprehensive survey of this literature see [4].

1.1. Multiple Solutions

Since geometric constraint systems correspond to nonlinear systems of equations, one difficulty that must be addressed is to select the "right" solution from a possible set that can be exponential in size for well-constrained problems. Here, the meaning of "right" depends strongly on the application. For example, if the constraint system is to express that two bodies should be mated on a common face plane, an acceptable solution would have to ensure that the bodies do not interpenetrate.

It is possible to express rigorously many of such application requirements, but this is not normally done for reasons of efficiency [3]. Rather, one relies on heuristics which in many cases succeed. When the heuristics fail, it becomes necessary to select a different solution, and this motivates finding, for subproblems that cannot be decomposed further, *all* possible solutions.

In earlier work we have investigated the case of planar constraint solving, as well as some spatial constraint problems; e.g., [7, 11, 12]. The spatial problems were restricted to a vocabulary of points and planes only, and involved only six primitives. In this paper, we investigate the larger class of problems involving lines as well.

1.2. Instance and Generic Solvers

We classify geometric constraint solvers by the manner in which they use the geometric information. In particular, the family of *instance solvers* solves a constraint problem using only specific instance information with fully valuated constraints. In contrast, *generic solvers* first analyze the constraint problem without regard of the specific constraint values, deriving a plan on how to decompose the problem and solve it subsequently. In a subsequent phase, the plan is then executed and the constraint problem instance is solved.

We advocate generic solvers because they allow the formulation of templates to solve entire classes of constraint problems, increasing solver efficiency. Moreover, interactive changes of dimensional constraint values allow recomputation with the same solution plan. In such cases, deriving the solution plan becomes a preprocessing step, another efficiency gain. In this paper, we look at the problem of how a generic solver would solve certain templates that arise in spatial constraint systems. The formulation of templates has been explored in [11] and [10].

2. Two Sample Problems

We illustrate the issues we are facing in spatial constraint solving by considering two simple configurations, each involving six primitives. The first such configuration consists of six points, with distance constraints between them in the topology of octahedron edges.

The problem has been analyzed before. In [22], the problem is studied in the context of kinematics of mechanisms and, in [6], as a problem in molecular design. In [11], it was studied using geometric reasoning and applying determinant theory.

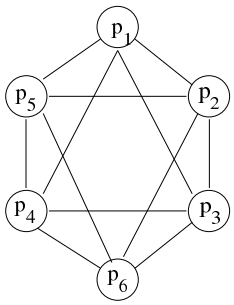

Figure 1. Octahedron Problem: Vertices $p_1, ..., p_6$ represent points, edges represent distance constraints.

We will show that this problem is approachable by a mix of standard symbolic and numerical techniques in a systematic framework. Furthermore, variations of the problem, in which some of the points are replaced with planes and some of the distance constraints with angle constraints, are also solvable with this generic approach.

The second problem to be considered also involves six primitives, but they are now three points and three lines. Between each pair of primitives there is a constraint, distance for point pairs and point/line pairs, and angle constraints between line pairs. Here we encounter significantly greater complexity.

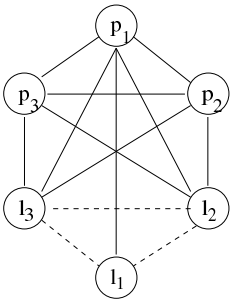

Figure 2. 3p3l Problem: Vertices p_1, p_2, p_3 represent points, vertices l_1, l_2, l_3 lines. Solid edges represent distance constraints, dashed edges angle constraints.

The proposed framework combines geometric reasoning with algebraic and analytical methods.

The geometric problem is translated into an *associated algebraic system* using an algebraic representation of the primitives. The associated algebraic system is reduced to a *core system* after symbolic simplification. The coefficients of the core system depend only on the constraint values. Moreover, the solutions of the constraint problem and the associated algebraic system can be computed easily from the solutions of the core system.

We find the solutions of the core system using homotopy continuation. Homotopy continuation has been applied successfully to problems in many areas, including robotics, kinematics of mechanisms, chemical equilibrium determination, geometric intersection [19, 29, 23, 26, 28, 14, 13] and, more recently, to constraint solving [15]. The method employs a predictor-corrector scheme to deform all the solutions of a known system, the *start system*, into the solutions of the system we want to solve (the *target system*). The process is also called *path tracking*, because it evaluates a *homotopy path* beginning at a solution of the start system, and the path may converge to a solution of the target system. Therefore, the number of paths to be tracked depends on the number of solutions of the start system. It is important to minimize the number of paths by choosing a good start system; [4].

We use two software packages, which implement different flavors of homotopy continuation: *Continuum* [5], which we implemented, uses a projective approach (also referred as standard homotopy) based on Bézout's theorem [18]. *PHC*, [28, 27], implements real homotopy continuation based on Bernstein's theorem. For more theory see [18, 26, 1, 16].

Since the coefficients of the system depend on the constraint values, we can further exploit the structure those parameters impose on the solution set by using parameter homotopy. Morgan and Sommese [21] have shown that, if the coefficients of a system are given by parameters, we can solve a system for generic set of parameter values (i.e. by standard homotopy), and then use that system and *only* its affine solutions in a homotopy to find all the affine solutions of any other system with the same parameter structure. Solving the generic system is now only preprocessing.

This technique is specially useful when a system has to be solved repeatedly for different parameter values, which is the case here. We have used Continuum for our experiments with parameter homotopy and have found that this reduces the number of paths significantly and so sharply reduces the computation time.

The representation of the primitives, simplification of the associated system, and the selection of the core system are driven by the following quantities which constitute a measure of the complexity of a system; [20, 19]:

1. Number of terms per equation,
2. Number of variables per equation,
3. Total degree (or Bezout bound),
4. Mixed volume (or BKK bound),
5. Numerical conditioning of the Jacobian matrices along the homotopy paths.

Intuitively, the number of terms per equation and the number of variables per equation provide a measure of the sparseness of the system. The total degree and the mixed volume give an upper bound on the number of solutions, and so give an intrinsic measure of the number of paths to be tracked by the homotopy method. Finally, well conditioned Jacobian matrices along the paths indicate that the numerical computation is stable.

Note that a system with fewer variables need not be a simpler system. Morgan [20] gives a number of examples where symbolic reduction degrades the numerical stability of a system, especially when the reduction is carried beyond a certain point. Therefore, a balance should be struck between seeking to reduce the number of variables, by symbolic algebraic computation and maintaining stable paths.

Finally, unless otherwise noted, all the running times reported in were obtained on a Sun Sparc Station 20 with 128MBytes of memory and SunOS 5.5.1.

3. The Octahedron Solved

We present the framework used to solve the the octahedron problem shown on Figure 1.

3.1. Representation

A point p_i is represented by its Cartesian coordinates

$$p_i : (x_i, y_i, z_i),$$

and the distance between two points p_i and p_j, by

$dist(p_i, p_j) = d_{ij}$	
vector	$\|\|p_i p_j\|\| = d_{ij}$
Cartesian	$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = d_{ij}^2$

3.2. Equation Formulation

The solutions of well-constrained problems are rigid realizations with six degrees of freedom (3 translational and 3 rotational). The remaining degrees of freedom will be eliminated by a suitable placement of some of the primitives.[1] We assume nonzero distances to avoid enumerating degenerate cases.

For the octahedron problem there is only one placement choice:

[1] This can also be thought of as choosing the coordinate systems such that (some of) the coordinates of those primitives are known in advance.

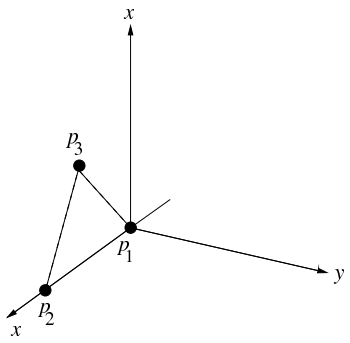

Figure 3. Placement of 3 points.

Placement of 3 Points:

Let p_1, p_2 and p_3 be 3 points with pairwise distance constraints $d_{i,j}$. A generic placement can be obtained by the following rules:

1. p_1 is placed at the origin.
2. p_2 is placed on the positive side of the x-axis at distance d_{12} from p_1.
3. p_3 is placed on the xz-plane according to the distances d_{13} and d_{23}, with $z \geq 0$.

See Figure 3. The points are now represented by

$$\begin{aligned} p_1 &: (0, 0, 0) \\ p_2 &: (x_0, 0, 0) \\ p_3 &: (x_1, 0, x_2) \end{aligned}$$

Moreover, the placement rules and the constraints completely determine the values of x_0, x_1 and x_2:

$$x_0 = d_{12} \quad x_1 = \frac{1}{2}\frac{x_0^2 - d_{23}^2 + d_{13}^2}{x_0} \quad x_2 = \sqrt{-x_1^2 + d_{13}^2}$$

The associated system, obtained in this way, has 12 equations and 12 variables. Furthermore its total degree and BKK bound equal 2^{12}. Therefore 4096 homotopy paths must be tracked to solve the system. Considering that each path is computed in 1 second, more than one hour would required to solve the problem.

3.3. Algebraic Simplification

The associated system can be simplified in the following steps (refer to [4] for the details):

1. Gaussian elimination. The resulting system should have as few squared variables as possible.
2. Eliminate univariate equations, since the variables involved can be determined directly.

3. Parameterize all bilinear equations and replace the occurrence of each parameterized variable, by the corresponding parametric expression.

4. Parameterize the bivariate quadratic equations (using sin and cos) and replace the occurrence of each parameterized variable, by the corresponding parametric expression.

5. Use the standard trigonometric substitution $\cos(\alpha_i) = \frac{1-y_i^2}{1+y_i^2}$, $\sin(\alpha_i) = \frac{2y_i}{1+y_i^2}$, where $y_i = \tan\left(\frac{\alpha_i}{2}\right)$.

The resulting core system

$$\begin{cases} (\alpha_1\, {y_2}^2 + \alpha_2)\, {y_1}^2 + \alpha_3\, y_2\, y_1 + \alpha_4\, {y_2}^2 + \alpha_5 = 0 \\ (\beta_1\, {y_3}^2 + \beta_2)\, {y_1}^2 + \beta_3\, y_3\, y_1 + \beta_4\, {y_3}^2 + \beta_5 = 0 \\ (\gamma_1\, {y_3}^2 + \gamma_2)\, {y_2}^2 + \gamma_3\, y_3\, y_2 + \gamma_4\, {y_3}^2 + \gamma_5 = 0 \end{cases} \quad (1)$$

has only 3 equations of degree 4. The coefficients α_i, β_i, and γ_i, $i = 1, \ldots, 5$ depend ultimately on the distance constraints. Furthermore, given a solution of the core system, a solution of the original system, and, consequently, a realization of the problem can be easily computed.

The total degree of the system 1 is 64 and its BKK bound is 16. Moreover, we used Continuum to solve various generic instances of system 1 and found that it has indeed 16 affine solutions.

3.4. Solution with Homotopy Continuation

Table 1 summarizes the application of homotopy continuation to a typical octahedron problem. Figures 4, 5, 6, and 4, show the realizations of a typical octahedron problem.

		Typical problem
Continuum Std. Homotopy	# paths	64
	time (in sec.)	27
Continuum Par. Homotopy	# paths	16
	time (in sec.)	2
PHC	# paths	16
	time (in sec.)	2
Solutions	Real	8
	Complex	8
	Geometric	8

Table 1. Summary of the results of the application of homotopy continuation on a typical octahedron problem.

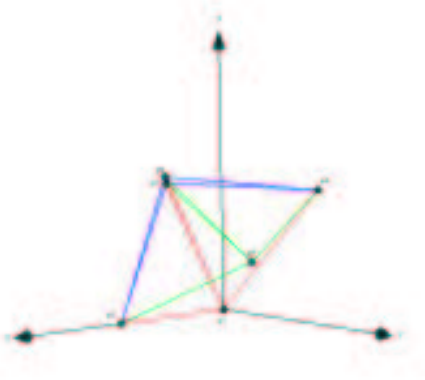

Figure 4. Realization #1 of a typical octahedron problem

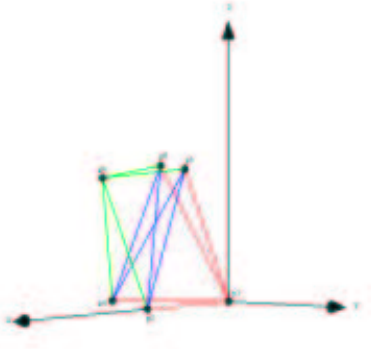

Figure 5. Realization #2 of a typical octahedron problem

4. The Points/Lines Problem

We consider now how to solve the 3p3l problem of Figure 2.

4.1. Representation

Points and distance between points are represented as defined in section 3. A line l_i is represented by the pair (b_i, t_i), where $b_i = (bx_i, by_i, bz_i)$ is the point on the line closest to the origin and $t_i = (tx_i, ty_i, tz_i)$, the unit tangent; i.e., $||t_i|| = 1$ and $b_i \cdot t_i = 0$:

$$l_i : (bx_i, by_i, bz_i : tx_i, ty_i, tz_i),$$

Based on this representation, we can define the equations associated with the following constraints:

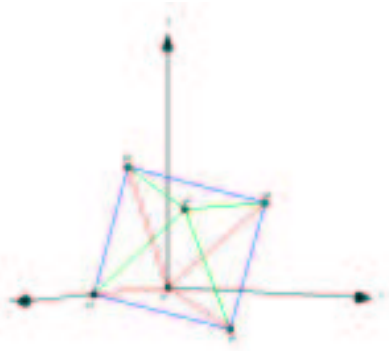

Figure 6. Realization #3 of a typical octahedron problem

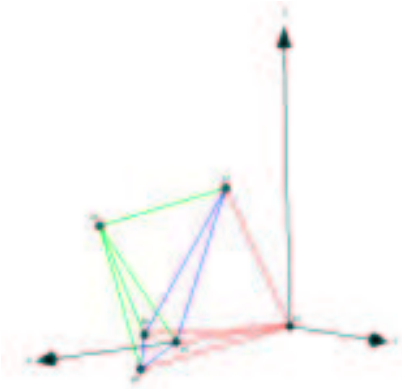

Figure 7. Realization #4 of a typical octahedron problem

Distance Point-Line

$dist(p_i, l_j) = d_{ij}$	
vector	$\|\|b_j - p_i\|\|^2 = d_{ij}^2 + < p_i, t_j >^2$
Cartesian	$(bx_i - x_i)^2 + (by_i - y_i)^2 (bz_i - z_i)^2 -$ $(x_i\, tx_i + y_i\, ty_i + z_i\, tz_i)^2 = d_{ij}^2$

Angle Line-Line

$ang(l_i, l_j) = a_{ij}$	
vector	$t_i \cdot t_j = \cos(a_{ij})$
Cartesian	$tx_i tx_j + ty_i ty_j + tz_i tz_j = \cos(a_{ij})$

We could represent lines alternatively with Grassmann-Plücker coordinates [25, 24]. However, we found that the resulting equations are more complex according to our criteria.

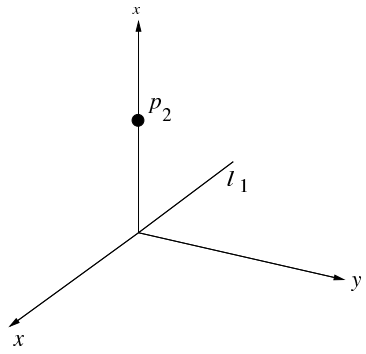

Figure 8. Placement of 1 point and 1 line

Blaschke [2] used dual quaternions to represent points, planes and lines, and to establish certain geometric constraints. Hestenes [8, 9] extends Blaschke's idea by using the concept of Geometric Algebras (also referred to as Clifford Algebras) to create a uniform representation for geometric primitives. Both approaches are attractive from a theoretical point of view, but must rely ultimately on the Euclidean or projective representations for the actual computation.

4.2. Equation Formulation

We assume nonzero distances and nontrivial angles ($\neq 0°, 180°$) to avoid enumerating degenerate cases.

For the 3p3l problem, there are three placement choices which yield to different associated systems. F_{ppp} is obtained by placing 3 points, as described in section 3. F_{pl} and F_{ll} are obtained, respectively, by placing one point and one line, or else two lines, as follows:

Placement of 1 Point and 1 Line: F_{pl}

Let $l_1 = (b_1, t_1)$ be a line, p_2 a point, and d_{12} the distance between l_1 and p_2. They can be placed as follows:

1. l_1 is put on the x-axis (with tangent vector $(1, 0, 0)$).

2. p_2 is placed on the positive side of the z-axis and at distance d_{12} from l_1.

See Figure 8.

For l_1 and p_2 we obtain

$$l_1 : (0, 0, 0 : 1, 0, 0)$$
$$p_2 : (0, 0, x_0)$$

where $x_0 = d_{12}$.

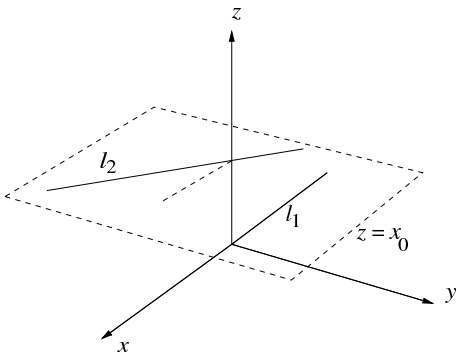

Figure 9. Placement of 2 lines

Placement of 2 Lines: F_{ll}

Let $l_1 = (b_1, t_1)$ and $l_2 = (b_2, t_2)$ be two lines, and let a_{12} be the angle between them. Then the lines can be placed as follows:

1. l_1 is placed as the x-axis (tangent $(1, 0, 0)$).
2. l_2 lies in the plane $z = x_0$, intersects the z axis, and satisfies the angle constraint a_{12}.

The situation is depicted on figure 9

The generic coordinates of l_1 and l_2 are

$$l_1 : (0, 0, 0 : 1, 0, 0)$$
$$l_2 : (0, 0, x_0 : x_1, x_2, 0)$$

The values of x_1 and x_2 depend only on the angle between the two lines,

$$x_1 = \cos(a_{12}) \quad x_2 = \sqrt{1 - x_1^2} = \sin(a_{12})$$

The value of x_0 cannot be determined in advance. It depends on the other primitives and the constraints of the problem.

4.3. Algebraic Simplification

How much each system can be simplified (according to our criteria) depends solely on its structure. Table 2 summarizes the simplification of the three systems of the different placement choices. The columns show the number of equations and the total degree of the systems before and after simplification. The simplification steps are detailed in [4]. The techniques included computation steps similar to those used in section 3, but also include steps familiar from Gröbner basis computations and other parameterizations.

Table 3 summarizes some of the complexity parameters for systems $\bar{F}_{ppp}$, $\bar{F}_{pl}$, and $\bar{F}_{ll}$, after simplification. These parameters are used to select the core system to be solved by homotopy continuation.

System	Before Simplification	
	# Eq.	Total degree
F_{ppp}	21	$2^{21} = 2097152$
F_{pl}	19	$2^{19} = 524288$
F_{ll}	18	$2^{18} = 262144$

System	After Simplification	
	# Eq.	Total degree
$\bar{F}_{ppp}$	9	$2^6 \times 6^3 = 13824$
$\bar{F}_{pl}$	16	$2^{16} = 65536$
$\bar{F}_{ll}$	12	$2^{12} = 4096$

Table 2. Summary of the simplification of the systems for the different placement choices.

System $\bar{F}_{pl}$ can be excluded, because it has a very high total degree and BKK bound. System $\bar{F}_{ppp}$ seems like the natural choice. Even though it has a large total degree, the BKK bound indicates that it may have fewer affine solutions than the other systems.

System	terms/equation	vars/equation
$\bar{F}_{ppp}$	$96/11 = 10.67$	$36/9 = 4$
$\bar{F}_{pl}$	$101/16 = 6.31$	$72/16 = 4.5$
$\bar{F}_{ll}$	$87/12 = 7.25$	$52/12 = 4.33$

System	Total degree	BKK bound
$\bar{F}_{ppp}$	$2^6 6^3 = 13824$	3456
$\bar{F}_{pl}$	$2^{16} = 65536$	6144
$\bar{F}_{ll}$	$2^{12} = 4096$	4096

Table 3. Complexity parameters of the simplified systems $\bar{F}_{ppp}$, $\bar{F}_{pl}$, and $\bar{F}_{ll}$.

In this case, however, the BKK bound overcounts the number of affine roots. Practical experiments solved system $\bar{F}_{ppp}$ for various (consistent) constraint values. It was found that 1904 out of the 3456 paths converge to affine solutions. Similar experiments performed on $\bar{F}_{ll}$ found that only 960 paths out of the 4049 paths lead to affine solutions. Therefore $\bar{F}_{ll}$ should be chosen as the core system of the 3p3l problem.

		Typical Example
Continuum Std. Homotopy	# paths	4096
	time	4h18m
Continuum Par. Homotopy	# paths	960
	time	1h34m
PHC	# paths	3456
	time (in sec.)	16h20m
Solutions	Real	48
	Complex	912
	Geometric	48

Table 4. Typical running times of Continuum and PHC for a 3p3l problem where the lines are orthogonal.

4.4. A Special Case

Even though $\bar{F}_{ll}$ is the simplest core system, it cannot be used to solve the 3p3l problem interactively, since there are still too many paths leading to affine solutions.

We analyzed a special case of the 3p3l problem in which the lines are pairwise orthogonal. The system that results is simpler, when considering the number of variables and terms per equation, but the bounds on the number of solutions remain the same. As in the general case, the system obtained by using different random distance values has 960 affine solutions.

4.5. Solution with Homotopy Continuation

Table 4 summarizes the running times of Continuum and PHC for a 3p3l problem with orthogonal lines. The best performance is achieved with parameter homotopy.

5. Discussion

We stated in the introduction that instance solvers, such as the Newton-Raphson solvers familiar from [17], are not well suited to explore alternative solutions. Moreover, they have very limited ability to decompose large constraint problems into a set of smaller ones that may be solved in isolation. While the latter problem can be remedied by using Newton iteration in the second phase of a generic solver, the former deficiency cannot be so addressed. This has motivated us to augment the possibility of solving template problems algebraically with the comprehensive numerical family of homotopy continuation methods.

With six geometric primitives, the 3p3l pattern explored in Section 4 can hardly be considered very large. Nevertheless, this template, and others involving line primitives, have so far resisted an adequate solution that is capable of determining all finite solutions. In our research, we have considered the nature of this template by exploring the following choices:

1. Different coordinatizations of the line primitive;
2. different ways to position the coordinate system, or, equivalently, a subset of primitives in the template;
3. a variety of symbolic algebraic techniques; and
4. several different ways to structure homotopy continuation solvers.

Unfortunately, none of these different ways to approach the 3p3l problem has resulted in a template solver sufficiently efficient to provide interactive solvers for those templates.

The implication of the technology barrier we have encountered is this: Allowing the application user to revisit the solution tree interactively and exploring other, perhaps more appropriate solution choices by choosing a different path through the tree, is not going to be sufficiently fast in practice. Thus, variational spatial constraint solvers built on a comprehensive, generic approach are not sufficiently attractive for interactive use.

We already mentioned that the formal incorporation of side conditions is unattractive, because it leads us to solving nonlinear optimization problems. Therefore, a way out of the apparent dilemma would be to severely restrict the vocabulary of spatial primitives. In particular, it appears that the use of lines as primitives must be restricted so as to allow interactive solvers that are capable of finding, in principle, all solutions of spatial constraints systems.

In the case of planar constraint problems, there are fast variational solvers based on restrictions that do not confine applications in CAD. In the case of spatial variational constraint solving, such pragmatic restrictions have not yet been demonstrated. It would be nice if technological progress would obviate the need to do so.

References

[1] E. Allgower and K. Georg. Continuation and Path Following. *Acta Numerica*, pages 1–64, 1993.

[2] W. Blaschke. *Kinematik und Quaternionen.* Berlin, Deutscher Verlag der Wissenschaften, 1960.

[3] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer Aided Design*, 27:487–501, 1995.

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

[5] C. Durand and C. M. Hoffmann. Continuum: A Homotopy Continuation Solver for Systems of Algebraic Equations. Technical Report TR 98-028, Department of Computer Sciences, Purdue University, 1998.

[6] I. Emiris and B. Mourrain. Polynomial System Solving and the Case of the Six-Atom Molecule. Technical Report 3075, INRIA, 1996.

[7] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *Intl. J. of Computational Geometry and Applications*, 6:405–420, 1996.

[8] D. Hestenes. *New Foundations for Classical Mechanics*. D. Reidel, 1987.

[9] D. Hestenes. The Design of Linear Algebra and Geometry. *Acta Applicandae Mathematicae*, 23:65–93, 1991.

[10] C. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming – CP97*, pages 463–477. Springer LNCS 1330, 1997.

[11] C. M. Hoffmann and P. J. Vermeer. Geometric constraint solving in R^2 and R^3. In D. Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, pages 266–298. World Scientific Publishing, 1994. second edition.

[12] C. M. Hoffmann and P. J. Vermeer. A spatial constraint problem. In *Workshop on Computational Kinematics*, France, 1995. INRIA Sophia-Antipolis.

[13] B. Huber. *Solving Sparse Polynomial Systems*. PhD thesis, Cornell University, 1996.

[14] B. Huber and B. Sturmfels. A Polyhedral Method for Solving Sparse Polynomial Systems. *Mathematics of Computation*, 64(212):1541–1555, October 1995.

[15] H. Lamure and D. Michelucci. Solving Geometric Constraints by Homotopy. In *Third Symposium on Solid Modeling and its Applications*, pages 263–269, Salt Lake City, Utah, 1995. ACM.

[16] T. LI. Numerical Solutions of Multivariate Polynomial Systems by Homotopy Continuation Methods. *Acta Numerica*, 6:399–436, 1997.

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

[18] A. Morgan. A Homotopy for Solving Polynomial Systems. *Applied Mathematics and Computation*, 18:87–92, 1986.

[19] A. Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[20] A. Morgan. Polynomial Continuation and its Relationship to the Symbolic Reduction of Polynomial Systems. In B. Donald, D. Kapur, and J. Mundy, editors, *Symbolic and Numerical Computation for Artificial Inteligence*. Academic Press, 1992.

[21] A. Morgan and A. Sommese. Coefficient-Parameter Polynomial Continuation. *Applied Mathematics and Computation*, 29:123–160, 1989.

[22] P. Nanua, K. Waldron, and V. Murthy. Direct Kinematic Solution of a Stewart Platform. *IEEE Transactions on Robotics and Automation*, 6(4):438–443, August 1990.

[23] N. Patrikalakis. Surface-to-Surface Intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, 1992.

[24] C. Springer. *Geometry and Analysis of Projective Spaces*. W.H. Freeman and Company, 1964.

[25] J. A. Todd. *Projective and Analytical Geometry*. Pitman Publishing Corporation, 1946.

[26] J. Verschelde. *Homotopy Continuation Methods for Solving Polynomial Systems*. PhD thesis, Katholieke Universiteit Leuven, 1996.

[27] J. Verschelde. PHCPACK, 1997. Available at http://www.math.msu.edu/~ jan/.

[28] J. Verschelde. PHCPACK: A general-purpose solver for polynomial systems by homotopy continuation. Technical Report TW 265, Department of Computer Science, Katholieke Universiteit Leuven, 1997.

[29] C. Wampler, A. Morgan, and A. Sommese. Numerical Continuation Methods for Solving Systems arising in Kinematics. *Journal of Mechanical Design*, 112:59–68, 1990.