Geometric Constraint Solving in Parametric Computer-Aided Design

Bernhard Bettig
Department of Mechanical Engineering,
West Virginia University,
Institute of Technology
Montgomery, WV 25136
e-mail: Bernhard.Bettig@mail.wvu.edu

Christoph M. Hoffmann
Department of Computer Science,
Purdue University,
West Lafayette, IN 47907
e-mail: cmh@cs.purdue.edu

With parametric computer-aided design (CAD) software, designers can create geometric models that are easily updated (within limits) by modifying the values of controlling parameters. These numeric and non-numeric parameters control the geometry in two ways: parametric operations and geometric constraint solving. This paper examines the advances over the last decade in the representation of parametric operations and of solving geometric constraint problems. An extensive literature has grown up surrounding geometric constraint solving and there has been substantial progress in the types of objects and constraints that can be handled robustly. Yet parametric operations have remained largely within the same conceptualization and begin to limit the flexibility of CAD systems, since they still do not align well with a systematic design process.
[DOI: 10.1115/1.3593408]

Keywords: computer-aided design, parametric CAD, geometric constraints, geometric constraint solving, variational solvers, declarative constraints

1 Introduction

Parametric computer-aided design (CAD) software is used pervasively in the design and manufacture of modern-day mechanical products. In parametric CAD, designers define the size, shape, and positions of geometric features and assembly components in terms of numerical and non-numerical parameters. By changing parameter values, a design can be easily modified, within limits.

Two computational mechanisms, intertwined in current parametric CAD software, are used to control the geometry from input parameters:

- parametric operations, such as *extrude* and *unite*, which construct geometric objects that satisfy implied constraints imposed when the user selects the operation and its inputs, and
- geometric constraint solving, which repositions and scales geometric objects in sketches and assemblies so that they satisfy constraints that are explicitly imposed on them by the user.

This paper surveys the state of the art in parametric operations and geometric constraint solving technology. It describes the state of the art a decade ago, as well as advances that have occurred since then.

The paper first addresses, in Sec. 2, the technical challenges and advances related to parametric operations in CAD and explains how geometric constraint solving fits with parametric CAD. Then, the bulk of the paper addresses advances in geometric constraint solving techniques, which are discussed in two major sections:

- In Sec. 3, a broad range of approaches to constraint solving is discussed. Over the years, many different approaches have been reported in the literature, and this section sketches them briefly.
- To date, the dominant approach to constraint solving is based on a constraint graph analysis that formulates a solution plan, followed by a solver that, usually recursively, elaborates the plan and computes a solution. This material is developed in Sec. 4.

Section 5 provides a summarizing discussion with some conclusions.

The scope of the paper is limited to the computation of the size, shape, and placement of geometric objects including points, curves (e.g., straight lines, arcs, conics, and freeform), and surfaces (e.g., planar, cylindrical, conical, spherical, and freeform) as controlled through implicitly or explicitly imposed geometric constraints. The constraints can be *dimensional*, relating CAD parameters to radius, distance, and angle, and they can be *geometric*, positing perpendicularity, concentricity, tangency, and so on. The problems can be posed in 2D or 3D space. The paper does not address the less centrally related subjects of geometric topology, feature semantics, knowledge-based engineering, or optimization.

2 Parametric Operations in CAD

Parametric operations are created when a designer uses solid modeling operations such as *extrude*, *unite*, and *blend*. The parametric operations are recorded as a sequence (or tree) of construction steps in a *construction history* that can be controlled by the user. Parameters, such as the radius for a blend operation, occur as inputs to the operations. If the value of a parameter is changed, the construction history is re-executed and the updated geometry is constructed, e.g., Ref. [1]. As shown by the examples in Table 1, parametric operations may include:

(1) user equations—controlling parameters from other parameters;
(2) parameter controlled geometry—controlling geometry from parameters and other geometry; and
(3) measured geometry—controlling parameters from geometric computations.

Variations of these operations may involve other types of parameters (e.g., logical, enumerated, and text string) and other geometric objects (e.g., complete sketches, data, and solids); however, the salient point is that there is a fixed directional dependency between

Contributed by the CAD/Solid Modeling Committee of ASME for publication in the Journal of Computing and Information Science in Engineering. Manuscript received September 27, 2010; final manuscript received March 10, 2011; published online June 14, 2011. Assoc. Editor: J. Shah.

Table 1 Examples of parametric operations versus constraint solving problems

Situation	Parametric operations	Constraint solving
User equations	1. $r_i := 5$ 2. $r_o := r_i + 5$	Parameters r_o, r_i $r_o - r_i = 5$ $r_o = 2ri$
Parameter controlled geometry	1. Parameter r_o:=5 2. Line L_1 := Line (1,0,0) 3. Line L_2 := Line (0,1,0) 4. Circle C_1 := Circle with radius tangent to two lines (r_o, L_1, L_2)	Parameter r_o Lines L_1, L_2 Circle C_1 $r_o = 10$ Fixed (L_1) Fixed (L_2) Tangent (L_1, C_1) Tangent (L_2, C_1) Radius (C_1, r_o)
Measured geometry	1. Point P_1 := Point (0,0,0) 2. Point P_2 := Point (10,0,0) 3. Parameter d := Distance Measure (P_1, P_2)	Points P_1, P_2 Parameter d Fixed (P_1) Fixed (P_2) Distance (P_1, P_2, d)

the input entities on the right-hand-side of the assignment operations and the output entities on the left-hand-side. Therefore, while the parametric operation conceptualization lends itself to the provision of computations with diverse kinds of geometric objects and (implied) constraints, the fixed directional dependency makes it necessary for users to plan ahead how the features of the model should be controlled and may require manually uncoupling relationships that would otherwise give rise to cyclic dependencies. This conceptualization contrasts with the constraint solving conceptualization in which there are no such fixed directional dependencies. Users introduce parameters and geometric objects first and then annotate them with constraints, which are then satisfied through constraint solving computations. Advances in parametric operations technology relate to improvements in the types and robustness of parametric operations themselves and to improvements in dealing with the inflexibility of the fixed directional dependencies.

2.1 Developments Until 2000. The earliest parametric CAD system dates from the 1970s [2]. It used a dual representation, describing solids both in constructive solid geometry (CSG) and in boundary-representation (B-Rep). In this dual representation, the solid model is represented as a binary tree in which the leaves are primitive shapes such as blocks and cylinders, and the branches are Boolean operations such as unite, intersect, and subtract. Each primitive can be thought of as a parametric operation with input parameters defining size and position. The output is then the shape of the primitive as a B-Rep. Boolean operations take selected B-Rep shapes as input and output the resulting B-Rep shape. The CSG tree provided a very basic construction history.

The first parametric CAD system, in the sense as it is understood today, is Pro-Engineer, released by Parametric Technology Corporation in the late 1980s [3]. In this CAD system, the sizes and positions of geometric objects could be directly related to each other [4]. Thus, if the user changed the value of a dimension between geometric objects, or moved a geometric object, this could initiate other geometric objects to be moved automatically. Dimensions and geometric constraints appeared in parametric operations in two ways. In the first way, dimensions appeared as inputs to parametric operations and specific constraints were implied. For example, in an *extrude* operation, an input dimension controlled the length of the extrusion and perpendicularity between the side faces and the extruded face was implied. In the second way, dimensions and geometric constraints could be annotated to curves on a 2D *sketch* that defined the cross-section profile for a sweep operation such as *extrude* or *revolve*. In a flurry of activity in the early 1990s, e.g., [5–7], it became standard for geometric constraint solving to be used for sketches.

In an effort to expand the benefits of constraint solving beyond the isolation of sketches in sweep operations, one commercial system (I-DEAS) implemented numeric equation solving capabilities with inequalities [5]. Some commercial systems (e.g., I-DEAS and CoCreate [3]) also implemented constraint solving for three-dimensional solid geometry. As well, some *feature modeling* research systems used constraint solving for computing feature sizes and locations in 3D [8]. One such system tries to maintain consistency between B-Rep, construction history, and feature model representations with constraint solving occurring independently within operations of the construction history [9]. Another system integrates a feature model with a cellular geometric model and performs all constraint solving prior to combining features in the cellular model [10]. However, none of these technologies have become wide-spread in CAD, likely due to the complexity of the interactions that are possible between the constraint solving and the parametric operations. On the other hand, 3D constraint solving is now commonplace in assembly modeling where component positioning constraints are satisfied only after the solid shapes have been generated from the parametric modeling operations.

Improvements in parametric operations themselves have primarily been in their robustness and variety, for example, edge blend operations that used to fail when the radius of the blend was greater than the width of the tangent face now automatically forgo the tangency requirement in order to obtain a solution (as shown in Fig. 1), see also Ref. [11]. However, the overall framework of parametric operations has stayed the same. Some applications have made interesting use of this framework, for example for parametric design optimization in which the dimensional parameters are manipulated by an optimization algorithm in order to satisfy a geometric goal (e.g., desired volume) or structural goal (e.g., desired stress from imposed loads [12]).

2.2 Developments Since 2000. The emphasis of the last decade has been on mitigating the rigidity inherent in parametric CAD owing to the parametric operations [13,14]. For example, it should be possible to simply grab a face and drag it to where it should be. Instead, the designer must find the controlling operation in the construction history and within that operation find the controlling parameter. This could be, for example, a dimension in a sketch or the length of an extrusion. Changing the value of the parameter may then have unintended side-effects in other operations, e.g., [1]. To overcome this inflexibility, a variety of hybrid modeling systems have been developed by vendors and researchers that combine parametric and direct-manipulation interfaces [15–17]. In these systems, it is possible to reposition faces, scale the sizes of features or even twist features. Unfortunately, these systems implement the direct modeling interactions as transformation operations that are simply added to the construction history as additional parametric operations. For example, using the Move Face operation in Siemens PLM NX 7.5 *Synchronous Technology* [18], the user can select a side face on a parametrically defined solid and translate it interactively in the graphics display to make the shape 2 mm wider. The magnitude and direction of the translation is recorded in a new parametric operation and the previous

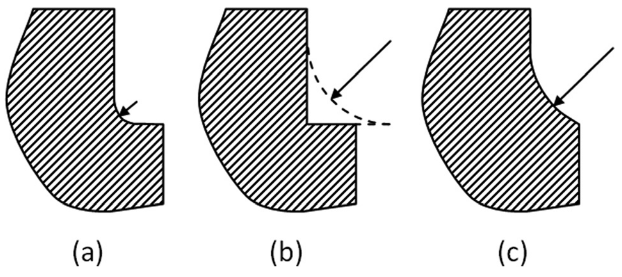

Fig. 1 Blended edge (*a*) small radius (*b*) error condition (*c*) tangency removed

operations in the construction history are maintained exactly as before. If the dimension for the original width is changed, the construction history is replayed, including the Move Face operation, which keeps the width 2 mm wider than specified. Thus, meaningful parametric control is lost. Another approach to resolve the inflexibility problem has been to treat some parameters as soft parameters and assign intervals [19] or set membership [20].

The limitations inherent with using parametric operations as a basis for design software are discussed by Bettig et al. [13]. They find that the search for design solutions when following a systematic design process is impeded by parametric operations because:

- Designs from parametric operations are implicitly fully constrained. In general, it is not clear which input parameters are controlled by requirements and which parameters can be tweaked. As well, some of the constraints implied by an operation may be superfluous with respect to the design intent or design requirements.
- Parametric operations are designed to output a unique solution. Often there are multiple mathematical solutions that should be explored.
- Parametric operations inherently bundle all implied constraints controlling an object into a single operation. Thus it is impossible to impose constraints from multiple sources onto a single object without manually combining them into one operation. It is also impossible to add further controls on an object once the object has been defined through a parametric operation. As well, coupled constraints must be manually uncoupled.
- The constraints or design intent implicit in a parametric operation can be violated by the parametric operations that follow it.

Future design software is proposed that does not rely on parametric operations; however, it is clear that parametric operations will continue to be used for the near future.

3 Major Approaches to Geometric Constraint Solving

The literature on geometric constraint solving often abstracts the problem as follows:

Given a set of geometric objects, such as points, lines, and circles, geometric and dimensional constraints, such as distance, tangency, perpendicularity etc., and an ambient space, usually the Euclidean plane assign coordinates to the geometric objects such that the constraints are satisfied or report that no such assignment has been found.

The *competence* of the solver is related to the report that no solution has been found: If no solution exists in that case, the solver is fully competent. On theoretical grounds, constraint solving is doubly exponential, so that in practice we settle for partial competence as long as the solver finds a solution for most of the problems arising in an application area, in acceptable time and space.

The main approaches to solving constraint problems are graph-based, logic-based, algebraic, and theorem prover-based. See also Ref. [21] that informs some of the material in this section.

3.1 Developments Until 2000

3.1.1 Graph-Based Approaches. In the graph-based approach, the constraint problem is translated into a labeled graph, the *constraint graph* with vertices representing the geometric objects that are constrained, and edges representing the constraints themselves. We distinguish three main strands: the *constructive approach*, the *degree of freedom* techniques, and *propagation* methods.

3.1.2 Constructive Approaches. In this approach, the constraint graph is decomposed and recombined to extract basic construction steps that must be solved. A second phase elaborates these steps, employing algebraic and/or numerical methods. This approach has become dominant and will be discussed in depth in Sec. 3.1.4.

3.1.3 Degrees of Freedom Analysis. The graph vertices are labeled with the number of degrees of freedom of the represented geometric object. In 2D, a point would have two degrees of freedom, a circle 3. Each graph edge is labeled by the degrees of freedom canceled by the represented constraints. If the incident vertices are points in 2D, for instance, an incidence constraint cancels two degree of freedom, a distance constraint cancels one degree of freedom. This graph is analyzed for a solution strategy.

Kramer [22,23] uses this approach to analyze and solve certain mechanisms. A symbolic solution method is derived using rules that have a geometric meaning. In Ref. [23], Kramer proves correctness of his method by establishing that the algorithm can be understood as a canonical rewrite system. Hsu and Brüderlin [24] solve the constraint problem in two phases, generating first a symbolic rules representation, followed by elaborating those rules by solving them. If geometric reasoning fails, a numerical solution is attempted.

Latham and Middleditch, [25], decompose the graph into minimal connected components they call *balanced sets*. If a balanced set is in a predefined set of patterns, then the subproblem is solved by a geometric construction, otherwise a numeric solution is attempted. This method also deals with symbolic constraints and identifies under- and overconstrained problems. Overconstrained problems are approached by prioritizing the given constraints.

3.1.4 Propagation Approaches. These methods encode the constraint problem by a graph in which the vertices represent variables and equations and edges are labeled with occurrences of variables in equations. Propagation seeks to orient the graph edges such that all incident edges to an equation vertex are incoming edges except for one. If this succeeds, then the equation system has been triangularized. Orientation algorithms include degree-of-freedom propagation and propagation of known values, e.g., Refs. [26,27]. The method fails when the orientation creates loops, so the algorithms include techniques to break loops [27] and may resort to numerical solvers. In Ref. [28], Borning et al. describe a local propagation algorithm that can deal with inequalities.

3.1.5 Logic-Based Approaches. In this approach, the constraint problem is translated into a set of geometric assertions and axioms. Applying geometric reasoning, this representation is transformed such that specific solution steps are made explicit. A set of construction steps is available to the solver and is solved by assigning appropriate coordinate values to the geometric entities.

Aldefeld [29], Brüderlin [30–32], Sohrt and Brüderlin [33], and Yamaguchi and Kimura [34] 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 [35] and Verroust et al. [36] consider two different types of constraints: sets of points placed with respect to a local coordinate frame and sets of straight line segments whose directions are fixed. 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, [37,38], 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.

3.1.6 Algebraic Approaches. In this approach, the problem is translated into a system of equations whose variables are the coordinates of the geometric elements and the equations express the constraints upon them. The equations are in general nonlinear. The main advantage of this approach is its completeness and dimension independence. A major difficulty of the approach is that the equation system is difficult to decompose into subproblems and that a general, complete solution of algebraic equations is inefficient. Note, however, that small algebraic systems arise in many of the other solution approaches and are routinely solved.

3.1.7 Symbolic Methods. General equation solvers employ symbolic techniques such as Gröbner bases [39] or the Wu-Ritt method [40,41] to triangularize the equation system. Buchanan and de Pennington [42] describe a solver built on top of the Buchberger's algorithm. Kondo reports a symbolic algebraic method in Ref. [43].

3.1.8 Numerical Methods. Numerical methods are among the oldest approaches to constraint solving. Numerical methods solve large systems of equations iteratively. Methods such as Newton iteration do well if a good approximation of the intended solution can be supplied and the system is not ill-conditioned. So, if the starting point is taken from the user's sketch, then the sketch should be close to the intended solution. Nonlinear systems have multiple solutions, but the numerical methods may find only one and may not offer control over the solution in which the user is interested.

Borning, [44], Hillyard and Braid, [45], and Sutherland, [46], use a relaxation method. This 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 *Newton-Raphson* iteration. It is used in the solvers described in Refs. [47–49]. 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 steps are taken, such as solving a least-squares problem.

Homotopy continuation, [50], is a family of methods that are global and guarantee convergence. They are exhaustive and allow to determine all solutions of a constraint problem. Their efficiency is worse than that of Newton-Raphson. Lamure and Michelucci, [51], and Durand, [52], apply this method to geometric constraint solving.

3.1.9 Theorem Proving. Solving a geometric constraint problem can be considered a subproblem of proving geometric theorems. However, 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's Wu-Ritt method, an algebraic-based geometric constraint solving method can be used to solve geometry theorems [41,53]. The method automatically finds necessary conditions to obtain nondegenerated solutions. In Ref. [40], Chou applies Wu's method to prove novel geometric theorems and [54,55] reports on automatic geometric theorem proving which allows to interpret, from a geometric point of view, the proof generated by computation.

3.2 Developments Since 2000.

Most of the key advances are described in Sec. 4. Here, we restrict to advances that interface with other areas or cannot be readily integrated into graph-constructive solvers.

3.2.1 Deformations. Deformation problems can be understood as constraint solving when there are restrictions placed on the type of deformation. For example, Moll and Kavraki [56] consider deformations that minimize bending energy, as does Ahn et al. [57] and others [58,59]. Surface deformation under area constraints, e.g., [60], also belongs in this category. These techniques and insights are rarely integrated with other geometric constraints such as distance from reference points, angle of intersection, perpendicularity, etc.

3.2.2 Dynamic Geometry. Given an underconstrained system, we can add constraints to make the problem well-constrained. These additional constraints can be understood as parameters when they are dimensional, and varying the parameter values, different solutions arise which can be collectively understood as a dynamic geometric configuration. A simple example would be a piston-crank assembly. Systems such as Cinderella [61] are designed to deal with such problems. A number of papers have investigated these problems from a constraint solving perspective, including Ref. [62].

3.2.3 Evolutionary Methods. In this approach, the problem is re-interpreted as an optimization problem that is attacked using genetic, particle-swarm or other evolutionary methods, e.g., [63–65].

4 Graph-Constructive Solvers

Graph-constructive solvers have become the dominant class of geometric constraint solvers.[1] This class of constraint solvers builds first a graph representing the constraint problem for the purpose of isolating specific, small subsets of geometric objects and constraints among them that can be solved separately. In a second phase, the solvers then recursively solve the actual constraints, guided by the graph decomposition, and determine coordinate assignments that solve the constraint problem. Each phase can end in failure, either because the constraints are not satisfiable or else because the solver does not succeed in breaking down the problem into subproblems that fit into the repertoire of subproblems the solver understands. In the following, we refer to the graph construction and analysis as *phase 1* of the solver and for the subsequent computations determining coordinates as *phase 2*. We can think of phase 1 as formulating a plan for solving the constraint problem and phase 2 as solving it according to this plan.

The graph that is analyzed in phase 1 has vertices representing the geometric objects to be instantiated and edges that represent constraints between them. Both vertices v and edges e are labeled with positive integral weights. The weight $w(v)$ of vertex v represents the *degrees of freedom* when placing the corresponding geometric object. For example, points and lines in the plane have two degrees of freedom. Put differently, the weight is equal to the number of independent coordinates of the geometric object. For edges $e=(v_1, v_2)$, the weight $w(e)$ is the number of coordinates of the adjacent vertices that can be determined from the equation expressing the constraint. For instance, if two points, represented by v_1 and v_2, are to be at a given distance, then $w((v_1, v_2))=1$, but if they are to be coincident, then $w((v_1, v_2))=2$.

The graph-constructive approach to constraint solving further divides into three families on account of whether the primary graph analysis is top-down, bottom-up, or hybrid. Additional distinctions can be drawn by the catalogue of graph patterns recognized by the graph analysis.

4.1 Developments Until 2000.

The top-down approach for 2D constraint problems was pioneered by Owen [6] in 1991. Owen recursively decomposes the constraint graph into triconnected components, in phase 1, searching for three vertices that split the graph into three subgraphs. In the recursive process, splitting the graph by an articulation node into two subgraphs corresponds to finding an under-constrained configuration in the constraint problem. The *triangles* found in this decomposition correspond to equation systems that involve solving univariate quadratic equations, thus are simple to solve.

The bottom-up approach was first proposed by Bouma et al. in 1993 and reported in Ref. [7]. Here, triangles are located in the graph and correspond to solvable subsystems, leading to the same repertoire of equation systems in phase 2 as in the top-down approach. Bottom-up solvers are good at determining over-constrained subproblems, both consistent and inconsistent. Both Owen's top-down and Bouma's bottom-up methods are of $O(n^2)$ complexity in phase 1 [66].

Research leading up to 2000 focused mainly on extending the repertoire of subgraphs that the bottom-up approach can handle and seeking good algorithms for solving the associated equation systems in phase 2. It also includes work that shores up the underlying theory of triangle solvers. In particular, we know that if there is a bottom-up decomposition, then any sequence of decomposition steps in phase 1 will succeed [67]. Moreover, the (variant of a) solution found in phase 2 does not depend on the order in

[1]We use this term in the broadest sense.

which phase 1 decomposed the graph: the same set of triples is interrogated, albeit in a different order [67].

Geometric constraint problems correspond to systems of nonlinear equations. Thus, a constraint problem can have multiple solutions. Which solution is intended is a difficult user-interface problem that was first broached in Ref. [7]. For the basic triangle decomposition solvers, the problem manifests in how to place three related geometric elements with respect to each other. So, Ref. [7] picks solutions in which such triples are placed as they were in the input sketch. This works well in many, but not in all, cases. Later work by Sitharam engages the user in a visual dialogue to obtain guidance from the user.

Extensions to the bottom-up solvers include variable-radius circles [6], certain cubic Bézier curves [68], conics [69], and subgraphs that involve solving algebraic equations that are higher than quadratic [7]. Owen's treatment of variable-radius circles is largely numerical. The equations that arise in general have high degree in some of the cases as discussed later.

Prior to 2000 there are also attempts at combining different approaches. Fudos and Hoffmann succeeded in combining top-down and bottom-up analysis in Ref. [70], so creating a hybrid solver. This allows dealing with under- and over-constrained problems uniformly. Moreover, Hoffmann and Joan-Arinyo [71] make a first cut at combining graph-constructive solvers with equation solvers opening the door to more general constraint problems that can use symbolic dimensional constraints and equations relating them by equations supplementing the geometric constraint specifications.

Graph-constructive solvers for spatial constraints are a natural next step and have been considered early-on. For spatial constraint solving using this approach, the main problem is to solve the arising subsystems of equations which are considerably more complex than in the planar case, even for very simple subgraph patterns. There are also many subgraph patterns needed for simple configurations if lines are allowed, a further barrier. Early work therefore restricts to points and planes in 3D.

In Ref. [72], Hoffmann and Vermeer begin exploring the basic subgraph patterns for spatial constraint solvers using points and planes only. The work explores both basic sequential as well as basic simultaneous configurations. The simplest nontrivial subgraph, for simultaneous problems, is the octahedron. In Ref. [73], this subgraph is considered and some of the cases are solved using geometric reasoning. Durand and Hoffmann [74,75] solves the equations of the octahedron using homotopy continuation. This allows a uniform approach to all arising cases but requires nontrivial numeric computation.

When allowing lines as part of the constraint problem, even sequential constructions can be complicated. For example, we can define a line in 3-space by its distance to four fixed points in space, asking effectively to find a common tangent to four given spheres. The associated equation system has degree 24, but with only 12 distinct solutions possible [76]. It can be shown that some problem instances have exactly 12 distinct solutions, thus establishing a tight bound on the number of common tangents.

Most of the work up to that point seeks to either extend the geometric vocabulary or identifying tractable and practically relevant subgraph patterns. But the possible subgraph patterns are infinite in number, so work begins before 2000 that asks whether there is a graph decomposition that does not restrict to a fixed set of subgraph patterns. Lomonosov and Sitharam begin this work together with Hoffmann and report a decomposition algorithm that identifies any solvable subgraph using a flow-based approach [77,78]. Sitharam perfects this algorithm later, as discussed below.

4.2 Developments Since 2000

4.2.1 Graph Decomposition. In a series of papers, Sitharam and collaborators complete the graph decomposition [79,80]. While earlier work concentrates on finding a subset of graph patterns that correspond to small, solvable subsets and are, at the same time, sufficiently general to have practical significance, Sitharam's *frontier algorithm* finds *all* subsets that correspond to subproblems solvable in isolation, thus generalizes the graph decomposition once and for all. Note that the frontier algorithm works for both 2D and 3D constraint problems, as well as for higher-dimensional spaces.

Contemporary work and later papers in this space work out variants of the algorithm or of the earlier decomposition algorithms that are easy to implement and improve specific details. For instance, Ref. [81] addresses the coupled decomposition when parametric constraints are present, Ref. [82] considers the domain of triangle decomposition, and Refs. [83,84] simplify the solver architecture. See also Ref. [85].

In 3D constraint solving, the number of simple patterns that can arise when allowing lines is very high, as discovered by Gao and his collaborators [86]. This means that graph constructive solvers must synthesize subgraph pattern as part of the graph decomposition. Thus, one aspect of the importance of Sitharam's algorithm is that the frontier algorithm does exactly that. But it also means that in phase 2, the algebraic equation systems will, in many cases, require generic techniques for solving, and that root selection also must be based on general principles. Gao's *locus intersection* method is one such approach [87].

Some of the problems associated with the 3D analysis constraint graphs involve characterizing rigidity. This problem has been addressed in papers by Sitharam and collaborators [88,89]. Mathis and Thierry [90] posit that the rigidity analysis of the decomposition/recombination approach captures problem invariance under rigid motion. He then extends the approach by considering other groups of geometric transformations, so deriving a more general view of decomposition and recombination.

4.2.2 Under- and Overconstrained Problems. Underconstrained and overconstrained problems may be amenable to special treatment. In the underconstrained case, Owen's top-down decomposition and Sitharam's frontier algorithm can pinpoint the subgraph that is incompletely constrained. More is possible when differentiating by constraint type. For instance, van der Meiden [91] identifies a type of subgraph where groups of angle constraints are recognized that lead to a finer decomposition and so allows better strategies for how to interact with the user to complete the constraints. In Ref. [92], he proposes nonrigid cluster rewriting configurations and techniques for root selection and for certain point configurations in 3D.

Overconstrained problems should be consistently overconstrained, for example, if a triangle is specified by three side lengths and one angle, then the angle value stipulated should be consistent with the required side lengths. Bottom-up solvers are well suited. General work on these problems includes the algorithm in Ref. [93] that addresses how to isolate overconstrained subgraphs.

Joan-Arinyo et al. [94] describe strategies to complete underconstrained problems. This also allows constraints to have priorities and originate from multiple views. This is an example of approaching overconstrained problems by grading the constraints, positing that some are more important than others. This approach is popular in applications. Jermann and Hosobe [95], so, approach overconstrained problems, allowing constraints to be arranged in hierarchies.

4.2.3 Variable-Radius Circles. For 2D constraint solvers, the triangle decompositions based on Refs. [6] and [7] provide a practical and useful subset of solvable problems. Extending this subset by variable-radius circles, that is, with circles whose radii are determined by the constraint configurations and are not explicitly given, are a logical extension that expands the solver competence significantly. For the graph analysis, two patterns must be added, one that determines center and radius from three constraints sequentially, the other in which the circle links with four constraints linking two clusters that can move relative to each other with one degree of freedom. While the sequential case is elementary, the second, simultaneous case yields algebraic equation systems that can be quite complex. The cases that arise have been investigated by Chiang and Hoffmann [96,97] and Chiang and Joan-Arinyo [98]. Owen already knew that one of the cases that arise in the sequential setting is the Apollonius problem. This case can be treated algebraically by transformation to a 3D configuration space in which the (up to eight) solutions are determined from univariate quadratic polynomials. The harder cases entail equation systems that must be solved numerically. Most recently, Chiang et al. revisit this problem and give a solution to the equation systems that exploits the parallelism of the graphics processing units (GPUs), so providing a fast and practical solution strategy; [99–101].

4.2.4 Valid Parameter Ranges. Given a constraint problem with dimensional constraints, we may ask what ranges of distance and angle constraints lead to solvable problems. This very difficult question has been considered in a number of papers [81,102,103]. In general, the problem requires restricting to individual parameters since the solution space is multidimensional and not necessarily connected, thus is difficult to explore. Joan-Arinyo and Mata [104] allow specifying intervals on dimensional parameters, and Mekhnacha et al. [105] allow applying probability distributions. Note that an exploration of valid parameter ranges in geometric constraint solving provides tools for tolerance and kinematic motion analyses.

4.2.5 Root Selection. There are two difficulties selecting, from the multiple solutions, one that corresponds to the application and user intent. The first difficulty is technical: what is a criterion for root selection that is invariant under translation and rotation. For triangle solvers one such criterion, used early-on, is a coordinate-free interpretation of the relative orientation of three geometric elements. The significance of Ref. [67] is that it shows the invariance of this criterion under alternative graph decompositions in phase 1. The second difficulty is that user guidance, for instance in CAD applications, is difficult to obtain because the solver is a deeply embedded component in CAD systems and the user is not likely to understand how the constraint problem has been formulated and how the solver works internally, thus posing questions to the user must be back-translated into terms that are visual and relate to the user's vocabulary. Sitharam et al. [106] guides user choice in her implementation by presenting the different root choices as graphical configurations of the shape elements.

Bettig and Shah [107] propose a set of inequality-based constraints such as *in front of/behind*, *on indicated side of*, *same orientation*, *concave/convex*, and *sharp/smooth* to allow users to specify the intended solution. Kale et al. [108] propose an addition to the frontier algorithm that inserts steps into the solution plan for checking the inequality conditions: if they are not met, backtrack through previous steps to obtain the next possible solution. The scheme has been found to be efficient as long as the inequality checking steps follow very closely behind the equation solving steps to which they apply.

4.2.6 Other Geometric Primitives in 2D. Gao et al. [109] discusses solving constraints with conics and linkages. There is some overlap between the constraint solving community and the CAGD community. Some work has attempted to fuse the two geometric vocabularies, to date with little practical impact. Examples of this work include [110]. Here, again, one key problem is that phase 2 has to work with equation systems of potentially high-degree. These difficulties can be overcome, in part, with GPU implementations [111].

Commercial solvers have been more conservative. They allow the traditional constraints on parametric curves such as prescribed end tangents, but they also allow constraints on curve length and maximum curvature [112].

4.2.7 3D Constraints. The early work was heavily influenced by graph pattern analysis. In contrast to 2D solvers, however, there seems to be no small subset that is both simple algebraically and practical in applications. In part, this is because 3D is inherently harder than 2D, but the difficulties may also be impacted by the paucity of 3D user interfaces and practical constraint patterns.

Restricting to points and planes, Michelucci [113] uses the Cayley-Menger determinant to devise an elegant algebraic solution to the octahedron pattern. Sitharam et al. [114] analyze coupled 2D and 3D systems as might arise in assemblies of variational parts. Gao et al. [86] show that the number of basic configurations numbers in the hundreds when lines are allowed as primitive geometric elements.

Geometry theorems constitute implicit constraints that may not be known to the solver. One method to expose hidden constraints is the *witness method* in which random configurations are investigated for unrecognized incidences and concurrencies, e.g., [61]. Michelucci and Foufou use the witness method to solve particular constraint problems, including 3D [115–117].

4.2.8 Numerical Methods. With the advent of arbitrarily complex subproblems, numerical solution methods need to be more competent. There is relatively little work on this aspect. Shi and Chen [118] consider the question and propose explicit techniques to isolate subproblems for subsequent numerical solutions. Gao et al. [86] propose to cut some of the constraints and mapping the problem to a dynamic geometry problem that is numerically approached. The resulting curves corresponding to the values of the cut constraints can then be intersected with the required values, so achieving a solution numerically.

4.3 Open Problems.

Geometric constraint solving has benefited from many theoretical and practical advances. Nonetheless, many open problems remain, both with respect to understanding the theoretical foundations as well as with respect to providing applications with better capabilities. We give a sampling of problems now.

It should be remembered that the graph analysis does not account for implicit relationships of the numerical parameters. Thus, even though a constraint graph may be analyzed as overconstrained, the parameter valuation may contain redundancies and the problem may well have a solution. Moreover, in 3D, a complete characterization of when a graph is well-constrained is not fully understood, e.g., [88–90].

Rigid subgraphs, identified by the graph analysis, can have arbitrary complexity, and therefore lead to algebraic equation systems of high degree. This *degree barrier* has been attacked with a variety of techniques, including most recently with GPU-based computations, e.g., [99–101]. These methods often rely on rendering the equations as manifolds and using graphics operations in the GPU to find potential solutions. When the dimensionality of those manifolds is high, straightforward GPU computations do not suffice and more abstract conceptualizations are necessary.

Given a well-constrained problem, finding valid parameter ranges is of great practical interest. Here, the difficulty is that the solution space is a complex, high-dimensional manifold that would be very costly to represent and map out fully. So, existing techniques consider small sets of parameters to keep the dimensionality low, e.g., [81,102–105]. Effective factorization theorems are needed to allow searching such restricted parameter sets in a way that yields information about the nature of the global solution range, from these local ones.

Equally of great practical interest is to find an effective strategy for root identification, i.e., to identify which of the different solution should be selected, e.g., [67,106–108]. As for the case of valid parameter ranges, the space of possible solutions is large and complex, so that mapping it completely is out of the question. Moreover, as parameters are varied, different paths through the solution tree may be necessary, depending on the user's application. In such a situation a semiautomated method should be considered, adding the difficulty of how best to communicate the consequence of choices in the interaction with the user.

Both in 2D and in 3D, it is desirable to seek incorporating additional shape primitives, and parametric curves and surfaces are an obvious choice, e.g., [110–112]. Here, we have a clash of conceptualizations: classical geometric constraint solvers ultimately solve algebraic equations, whereas in CAGD many of the degrees of freedom are determined by control points or knots. A unification of the two bodies of work may well require a radically different approach. Similarly, specifying constraints of minimum length or bending energy is of practical interest, yet little is known about incorporating such constraints on curves and surfaces into geometric constraint problems [57].

5 Conclusions

Over the last decade, the fundamental representations underlying parametric control of geometry in CAD systems has changed little. Most of the advances have been to expand the types of objects and constraints that are recognized and can be handled robustly. However it has also been recognized that the one-way dependencies inherent in parametric operations severely limit the flexibility of parametric CAD for designers and cause it to map poorly to recognized systematic design processes. This observed inflexibility offers opportunities for future break-throughs.

Geometric constraint solving methods have been developed over the last decade to expand significantly the scope of solvers, both in regards to the constraint graph structure analysis (phase 1 of Sec. 4), as well as the types of geometric primitives allowed. The expanded vocabulary is in part the result of new insights into geometric properties, and in part reflects advances in solver software. In particular, GPU-based equation solvers have been shown to make formerly difficult subproblems easy through sampling and parallelization. Graphics coprocessors will have continued impact going forward, but their use should include more abstracted techniques as explained.

To-date, graph-based constraint solving continues to dominate. The beginning part of the decade saw the achievement of a general understanding of graph-based solving that allowed it to be broadly applied. Advances in graph-based solving have also been key with respect to specific subchallenges, including dealing with under- and over-constrained problems, variable radius circles, identifying valid parameter ranges, root selection techniques, and 3D constraints. Nevertheless, there are open problems despite these achievements.

Some new approaches have also been developed, for example using evolutionary algorithms. The use of techniques from dynamic geometry and the consideration of deformable geometric objects are more examples of attempts to broaden the scope of geometric constraint solving. These advances testify to a vigorous research field with many applications outside CAD as well.

Acknowledgment

This work was supported in part by the National Science Foundation by Grants 0722210 and 0938999 and by a gift from the Intel Corporation.

References

[1] Hoffmann, C. M., and Joan-Arinyo, R., 2002, "Parametric modeling," *Handbook of CAGD*, G. Farin, J. Hoschek, and M.-S. Kim, Eds., Elsevier, New York, pp. 519–541.
[2] Requicha, A. A. G., 1980, "Representations for Rigid Solids: Theory, Methods, and Systems," ACM Comput. Surv., **12**, pp. 437–464.
[3] Parametric Technology Corp. Cocreate, 2010, www.ptc.com/products/cocreate
[4] Hoffmann, C. M., 2005, "Constraint-Based Computer-Aided Design," ASME J. Comput. Inf. Sci. Eng., **5**, pp. 182–187.
[5] Chung, J., and Schussel, M., 1990, "Technical Evaluation of Variational and Parametric Design," Comput. Eng., **1**, pp. 289–298.
[6] Owen, J. C., 1991, "Algebraic Solution for Geometry from Dimensional Constraints," *ACM Symposium Foundations of Solid Modeling*, pp. 397–407.
[7] Bouma, W., Fudos, I., Hoffmann, C. M., Cai, J., and Paige, R., 1995, "A Geometric Constraint Solver," Comput.-Aided Des., **27**, pp. 487–501.
[8] Shah, J., and Mantyla, M., 1995, *Parametric and Feature-Based CAD/CAM*, Wiley, New York, NY.
[9] Venkatamaran, S., 2000, "Integration of Design by Features and Feature Recognition," Master's thesis, Arizona State University.
[10] Bidarra, R., 1999, "Validity Maintenance in Semantic Feature Modeling," PhD thesis, Technische Universiteit Delft.
[11] Braid. I., 1996, "Non-Local Blending of Boundary Models," Comput.-Aided Des., **29**, pp. 89–100.
[12] Hardee, E., Chang, K.-H., Tu, J., Choi, K. K., Grindeanu, I., and Yu, X., 1999, "A CAD-Based Design Parameterization for Shape Optimization of Elastic Solids," Adv. Eng. Software, **30**, pp. 185–199.
[13] Bettig, B., Bapat, V., and Bharadwaj, B., 2005, "Limitations of Parametric Operators for Supporting Systematic Design," in *Proceedings of ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, DETC2005.
[14] Ilies, H. T., 2006, "Parametric Solid Modeling," in *Proceedings of the ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, DETC2006.
[15] Clarke, C., 2009, "Super Models," Engineer, **294**, pp. 36–38.
[16] Samuel, S., 2006, "CAD Package Pumps up the Parametrics," Mach. Des., **78**, pp. 82–84.
[17] Wu, N., and Ilies, H., 2007, "Motion-Based Shape Morphing of Solid Models," in *Proceedings of the ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, IDETC2007.
[18] Siemens PLM Software. Synchronous Technology, 2011.
[19] Wang, Y., 2007, "Solving Interval Constraints by Linearization in Computer-Aided Design," Reliab. Comput., **13**, pp. 211–244.
[20] Nahm, Y.-E., and Ishikawa, H., 2006, "A New 3D-CAD System for Set-Based Parametric Design," Int. J. Adv. Manuf. Technol., **29**, pp. 137–150.
[21] Hoffmann, C. M., and Joan-Arinyo, R., 2005, "A Brief on Constraint Solving," Comput.-Aided Des., **2**, pp. 655–663.
[22] Kramer, G. A., 1991, "Using Degree of Freedom Analysis to Solve Geometric Constraint Systems," *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, J. Rossignac and J. Turner, eds., pp. 371–378.
[23] Kramer, G. A., 1992, *Solving Geometric Constraint Systems*, MIT, Cambridge.
[24] Hsu, C.-Y., and Brüderlin, B., 1997, "A Hybrid Constraint Solver Using Exact and Iterative Geometric Constructions," *CAD Systems Development: Tools and Methods*, D. Roller and P. Brunet, eds., Springer-Verlag, Berlin, 1997, pp. 266–298.
[25] Latham, R., and Middleditch, A., 1996, "Connectivity Analysis: A Tool for Processing Geometric Constraints," Comput.-Aided Des., **28**, pp. 917–928.
[26] Freeman-Benson, B., Maloney, J., and Borning, A., 1990, "An Incremental Constraint Solver," Commun. ACM, **33**, pp. 54–63.
[27] Veltkamp, R., and Arbab, F., 1992, "Geometric Constraint Propagation With Quantum Labels," in Eurographics Workshop on Computer Graphics and Mathematics, pp. 211–228.
[28] Borning, A., Anderson, R., and Freeman-Benson, B., 1996, "Indigo: A Local Propagation Algorithm for Inequality Constraints, in ACM UIST '**96**, pp. 129–136.
[29] Aldefeld, B., 1998, "Variation of Geometric Based on a Geometric-Reasoning Method," Comput.-Aided Des., **20**, pp. 117–126.
[30] Brüderlin, B. D., 1988, "Rule-Based Geometric Modelling," PhD thesis, Institut für Informatik der ETH Zürich.
[31] Brüderlin, B. D., 1990, "Symbolic Computer Geometry for Computer Aided Geometric Design," *Advances in Design and Manufacturing Systems* pp. 177–181.
[32] Brüderlin, B. D., 1993, Using Geometric Rewrite Rules for Solving Geometric Problems Symbolically," *Theoretical Computer Science* **116**, pp. 291–303.
[33] Sohrt, W., and Brüderlin, B. D., 1991, Interaction with Constraints in 3D Modeling. Int. J. Comput. Geom. Appl., **1**, pp. 405–425.
[34] Yamaguchi, Y., and Kimura, F., 1990, "A Constraint Modeling System for Variational Geometry," *Geometric Modeling for Product Engineering*, J. U. Turner, M. J. Wozny, and K. Preiss, eds., Elsevier, North Holland, pp. 221–233.
[35] Sunde, G., 1987, "A CAD System With Declarative Specification of Shape," Eurographics Workshop on Intelligent CAD Systems, pp. 90–105.
[36] Verroust, A., Schonek, F., and Roller, D., 1992, "Rule-Oriented Method for Parameterized Computer-Aided Design," Comput.-Aided Des., **24**, pp. 531–540.
[37] Joan-Arinyo, R., and Soto, A., 1997, "A Correct Rule-Based Geometric Constraint Solver," Comput. Graphics, **21**, pp. 599–609.
[38] Joan-Arinyo, R., and Soto, A., 1997, "A Ruler-and-Compass Geometric Constraint Solver," *Product Modeling for Computer Integrated Design and Manufacture*, M. J. Pratt, R. D. Sriram, and M. J. Wozny, eds., pp. 384 – 393.
[39] Buchberger, B., 1985, "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory," *Multidimensional Systems Theory*, D. Reidel Publishing, pp. 184–232.
[40] Chou, S.-C., 1988, "An Introduction to Wu's Method for Mechanical Theorem Proving in Geometry," J. Automated Reasoning, **4**, pp. 237–267.
[41] Wu, W.-T., 1994, "Mechanical Theorem Proving in Geometries," *Texts and Monographs in Symbolic Computations*, B. Buchberger and G. E. Collins, eds., Springer-Verlag, Berlin.
[42] Buchanan, S. A., and de Pennington, A., 1993, "Constraint Definition System: A Computer-Algebra Based Approach to Solving Geometric-Constraint Problems," Comput.-Aided Des., **25**, pp. 741–750.
[43] Kondo, K., 1992, "Algebraic Method for Manipulation of Dimensional Relationships in Geometric Models," Comput.-Aided Des., **24**, pp. 141–147.
[44] Borning, A., 1981, "The Programming Language Aspects of ThingLab, a Constrained Oriented Simulation Laboratory," ACM Trans. Program. Lang. Syst., **3**, pp. 353–387.

[45] Hillyard, R., and Braid, I., 1978, "Characterizing Non-Ideal Shapes in Terms of Dimensions and Tolerances," Proceedings of ACM Computer Graphics, pp. 234–238.
[46] Sutherland, I., 1963, "Sketchpad, a Man-Machine Graphical Communication System," in *Proceedings of the Spring Joint Computing Conference*, IFIPS, pp. 329–345.
[47] Light, R., and Gossard, D., 1982, "Modification of Geometric Models Through Variational Geometry, Comput.-Aided Des., **14**, pp. 209–214.
[48] Lin, V. C., Gossard, D. C., and Light, R. A., 1981, "Variational Geometry in Computer-Aided Design," ACM Comput. Graphics, **15**, pp. 171–177.
[49] Nelson, G., 1985, "Juno, a Constraint-Based Graphics System. SIGGRAPH, pp. 235–243.
[50] Allgower, E., and Georg, K., 1993, "Continuation and Path Following," Acta Numerica, **7**, pp. 1–64.
[51] Lamure, H., and Michelucci, D., 1995, "Solving Geometric Constraints by Homotopy," Third Symposium on Solid Modeling and Applications, C. M. Hoffmann and J. Rossignac, eds., pp. 263–269.
[52] Durand, C., 1998, "Symbolic and Numerical Techniques for Constraint Solving," PhD thesis, Computer Science, Purdue University.
[53] Wu, W.-T., 1986, "Basic Principles of Mechanical Theorem Proving in Geometries," J. Syst. Sci. Math. Sci., **4**, pp. 207–235.
[54] Chou, S.-C., Gao, X.-S., and Zhang, J.-Z., 1996, "Automated Generation of Readable Proofs With Geometric Invariants: Multiple and Shortest Proof Generation," J. Automat. Reason. **7**, pp. 325–347.
[55] Chou, S.-C., Gao, X.-S., and Zhang, J.-Z., 1996, "Automated Generation of Readable Proofs With Geometric Invariants: Theorem Proving With Full Angles," J. Automat. Reason., **7**, pp. 349–370.
[56] Moll, M., and Kavraki, L., 2006, "Path Planning for Deformable Linear Objects," IEEE J. Rob., **22**, pp. 625–636.
[57] Ahn, Y. J., Hoffmann, C. M., and Rosen, P., 2011, "Length and Energy of Quadratic Bézier Curves and Applications" (submitted).
[58] Bao, F., Sun, Q., Pan, J., and Duan, Q., 2010, "A Blending Interpolator With Value Control and Minimal Strain Energy," Comput. Graphics, **34**, pp. 119–124.
[59] Ginkel, I., and Umlauf, G., 2008, "Local Energy-Optimizing Subdivision Algorithms," Comput. Aided Geom. Des., **25**, pp. 137–147.
[60] Xu, Y., Joneja, A., and Tang, K., 2009, "Surface Deformation Under Area Constraints," Comput. Aided Geom. Des., **6**, pp. 711–719.
[61] Kortenkamp, U., and Richter-Gebert, J., 2010, *The Interactive Geometry Software Cinderella.2*, Springer-Verlag, Berlin.
[62] Freixas, M., Joan-Arinyo, R., and Soto-Riera, A., 2008, "A Constraint-Based Dynamic Geometry System," ACM Solid and Physical Modeling, pp. 37–46.
[63] Cao, C., Zhang, B., Wang, L., and Li, W., 2006, "The Parametric Design Based on Organizational Evolutionary Algorithm," in PRICAI 2006 – 9th Pacific Rim International Conference on Artificial Intelligence, pp. 940–944, Springer Lect. Notes in AI 4099.
[64] Yuan, H., Li, W., Yi, R., and Zhao, K., 2006, "The TPSO Algorithm to Solve Geometric Constraint Problems," Comput. Inform. Syst., **2**, pp. 1311–1316.
[65] Gao, X.-Y., Sun, L.-Q., and Sun, D.-S., 2009, "Artificial Immune-Chaos Hybrid Algorithm for Geometric Constraint Solving," Inf. Technol. J., pp. 360–365.
[66] Fudos, I., 1995, "Constraint Solving for Computer Aided Design," PhD thesis, Purdue University, Department of Computer Sciences.
[67] Fudos, I., and Hoffmann, C. M., 1996, "Correctness Proof of a Geometric Constraint Solver," Int. J. Comput. Geom. Appl., **6**, pp. 405–420.
[68] Hoffmann, C. M., and Peters, J., 1995, "Geometric Constraints for CAGD," *Mathematical Methods for Curves and Surfaces*, M. Daehlen, T. Lyche, and L. Schumaker, eds., Vanderbilt University Press, 1995, pp. 237–254.
[69] Fudos, I., and Hoffmann, C. M., 1996, "Constraint-Based Parametric Conics for CAD," Comput.-Aided Des., **28** 91–100.
[70] Fudos, I., and Hoffmann, C. M., 1997, "A Graph-Constructive Approach to Solving Systems of Geometric Constraints," ACM Trans. Graphics, **16**, pp. 179–215.
[71] Hoffmann, C. M., and Joan-Arinyo, R., 1997, "Symbolic Constraints in Constructive Geometric Constraint Solving," J. Symb. Comput., **23**, pp. 287–300.
[72] Hoffmann, C. M., and Vermeer, P. J., 1994, "Geometric Constraint Solving in R^2 and R^3," *Computing in Euclidean Geometry*, 2nd ed., D. Z. Du and F. Hwang, eds., World Scientific Publishing, Singapore, pp. 266–298.
[73] Hoffmann, C. M., and Vermeer, P. J., 1995, "A Spatial Constraint Problem," *Computational Kinematics*, J.-P. Merlet and B. Ravani, eds., Kluwer Acad. Publ., pp. 83–92.
[74] Durand, C., and Hoffmann, C. M., 1999, "Variational Constraints in 3D," in *Proceedings of International Conference on Shape Modeling and Applications*, pp. 90–97.
[75] Durand, C., and Hoffmann, C. M., 2000, "A Systematic Framework for Solving Geometric Constraints Analytically," J. Symb. Comput., **30**, pp. 493–520.
[76] Hoffmann, C. M., and Yuan, B., 2000, "On Spatial Constraint Solving Approaches," Proceedings of ADG 2000, ETH Zurich, in press.
[77] Hoffmann, C. M., Lomonosov, A., and Sitharam, M., 1997, "Finding Solvable Subsets of Constraint Graphs," *Principles and Practice of Constraint Programming – CP97*, Springer LNCS 1330, NY, pp. 463–477.
[78] Hoffmann, C. M., Lomonosov, A., and Sitharam, M., 1998, "Geometric Constraint Decomposition," *Geometric Constraint Solving and Applications*, B. Bruderlin and D. Roller, eds., pp. 170–195.
[79] Hoffmann, C. M., Lomonosov, A., and Sitharam, M., 2001, "Decomposition Plans for Geometric Constraint Problems, Part I: Performance Measures for CAD," J. Symb. Comput., **31**, pp. 367–408.
[80] Hoffmann, C. M., Lomonosov, A., and Sitharam, M., 2001, "Decomposition Plans for Geometric Constraint Problems, Part II: New Algorithms," J. Symb. Comput., **31**, pp. 409–428.
[81] Joan-Arinyo, R., and Soto-Riera, A., 1999, "Combining Constructive and Equational Geometric Constraint Solving Techniques," ACM Trans. Graphics, **18**, pp. 35–55.
[82] Joan-Arinyo, R., Soto-Riera, A., Vila-Marta, S., and Vilaplana, J., 2001, "On the Domain of Constructive Geometric Constraint Solving Techniques," *IEEE Spring Conference on Computer Graphics*, R. Duricovic and S. Czanner, eds., Budmerice, Slovakia, pp. 49–54.
[83] Joan-Arinyo, R., Soto-Riera, A., Vila-Marta, S., and Vilaplana, J., 2002, "Declarative Characterization of a General Architecture for Constructive Geometric Constraint Solvers," *The Fifth International Conference on Computer Graphics and Artificial Intelligence*, D. Plemenos, ed., Limoges, France, pp. 63–76.
[84] Joan-Arinyo, R., Soto-Riera, A., Vila-Marta, S., and Vilaplana, J., 2004, "Revisiting Decomposition Analysis of Geometric Constraint Graphs," Comput.-Aided Des., **36**, pp. 123–140.
[85] Jermann, C., Trombettoni, G., Neveu, B., and Mathis, P., 2006, "Decomposition of Geometric Constraints Systems: A Survey," Int. J. Comput. Geom. Appl., **23**, pp. 1–35.
[86] Gao, X.-S., Hoffmann, C. M., and Yang, W., 2002, "Solving Spatial Basic Geometric Constraint Configurations With Locus Intersection. *Solid Modeling '02*, pp. 95–104.
[87] Gao, X.-S., Hoffmann, C. M., and Yang, W., 2004, "Solving Spatial Basic Geometric Constraint Configurations With Locus Intersection," Comput.-Aided Des., **36**, pp. 111–122.
[88] Sitharam, M., and Zhou, Y., 2004, "A Tractable, Approximate Characterization of Combinatorial Rigidity in 3D," in *5th Automated Deduction in Geometry*.
[89] Gao, H., and Sitharam, M., 2008, "Characterizing 1-dof Henneberg Graphs With Efficient Configuration Spaces," arXiv:0810.1997v2.
[90] Mathis, P., and Thierry, S., 2010, "A Formalization of Geometric Constraint Systems and Their Decomposition," Formal Aspects of Computing, **22**, pp. 129–151.
[91] van der Meiden, H., 2008, "Semantics of Families of Objects," PhD thesis, Delft University of Technology, Netherlands.
[92] van der Meiden, H., and Bronsvoort, W., 2010, "A Non-Rigid Cluster Rewriting Approach to Solve Systems of 3D Geometric Constraints," Comput.-Aided Des., **42**, pp. 36–49.
[93] Hoffmann, C. M., Sitharam, M., and Yuan, B., 2004, "Making Constraint Solvers Useable: Overconstraints, Comput.-Aided Des., **36**, pp. 377–399.
[94] Joan-Arinyo, R., Soto-Riera, A., and Vilaplana-Pastó, M., 2003, "Transforming an Underconstrained Geometric Constraint Problem into a Well-Constrained One," *Symposium on Solid Modeling and appl.*, pp. 33–44.
[95] Jermann, C., and Hosobe, H., 2008, "A Constraint Hierarchies Approach to Geometric Constraint Sketches," *23rd SAC '08*, pp. 1843–1844.
[96] Chiang, C.-S., and Hoffmann, C. M., 2001, "Variable-Radius Circles in Cluster Merging, Part I: Translational Clusters," Comput.-Aided Des., **34**, pp. 787–797.
[97] Chiang, C.-S., and Hoffmann, C. M., 2001, "Variable-Radius Circles in Cluster Merging, Part II: Rotational Clusters," Comput.-Aided Des., **34**, pp. 799–805.
[98] Chiang, C.-S., and Joan-Arinyo, R., 2004, "Revisiting Variable-Radius Circles in Constructive Geometric Constraint Solving," CAGD, **221**, pp. 371–399.
[99] Hoffmann, C. M., Chiang, C.-S., and Rosen. P., 2010, "Hardware Assist for Constrained Circle Constructions I," Comput.-Aided Des. Appl, **7**, pp. 17–33.
[100] Hoffmann, C. M., Chiang, C.-S., and Rosen, P., 2010, "Hardware Assist for Constrained Circle Constructions II," Comput.-Aided Des. Appl, **7**, pp. 33–44.
[101] Chiang, C.-S., Hoffmann, C. M., and Rosen. P., "A Generalized Malfatti Problem. *Computational Geometry Theory and Applications* (in press).
[102] Hoffmann, C. M., and Kim, K.-J., 2001, "Towards Valid Parametric CAD models," Comput.-Aided Des., **33**, pp. 81–90.
[103] van der Meiden, H., and Bronsvoort, W., 2006, "A Constructive Approach to Calculate Parameter Ranges for Systems of Geometric Constraints," Comput.-Aided Des., **38**, pp. 275–283.
[104] Joan-Arinyo, R., and Mata, N., 2001, "Applying Constructive Geometric Constraint Solvers to Geometric Problems With Interval Parameters," Nonlinear Anal. Theory, Methods Appl., **47**, pp. 213–224.
[105] Mekhnacha, K., Mazer, E., and Bessiere, P., 2001, "The Design and Implementation of a Bayesian CAD Modeler for Robotic Applications," Adv. Rob., **15**, pp. 45–69.
[106] Sitharam, M., Arbree, A., Zhou, Y., and Kohareswaran, N., 2006, "Solution Management and Navigation for 3d Geometric Constraint Systems," ACM TOG, **25**, pp. 194–213.
[107] Bettig, B., and Shah, J., 2003, "Solution Selectors: A User-Oriented Answer to the Multiple Solution Problem in Constraint Solving," ASME J. Mech. Des., **125**, pp. 443–451.
[108] Kale, V., Bettig, B., and Bapat, V., 2008, "Geometric Constraint Solving With Solution Selectors," In *Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DETC2008.
[109] Gao, X.-S., Jiang, K., and Zhu. C.-C., 2002, "Geometric Constraint Solving With Conics and Linkages," Comput.-Aided Des., **34**, pp. 421–433.
[110] Cheteut, V., Daniel, M., Hahmann, S., LaGreca, R., Lon, J., Maculet, R., and Sauvage, B., 2007, "Constraint Modeling for Curves and Surfaces in CAGD," Intl. J. Shape Model., **13**, pp. 159–199.
[111] Ahn, Y.-J., and Hoffmann, C. M., 2010, "Constraint-Based ln-Curves," SAC, pp. 1242–1246.

[112] Hanniel, I., and Haller, K., 2009, "Solving Global Geometric Constraints on Free-Form Curves," In *ACM Symposium Solid and Physics Modeling*, pp. 307–312.
[113] Michelucci, D., 2004, "Using Cayley Menger determinants," In *Proceedings of the 2004 ACM Symposium on Solid Modeling*, pp. 285–290.
[114] Sitharam, M., Oung, J., Arbree, A., and Zhou, Y., 2006, "Mixing Features and Variational Constraints in 3d," Comput.-Aided Des., 38.
[115] Foufou, S., Michelucci, D., and Jurzak, J.-P., 2005, "Numerical Decomposition of Geometric Constraints," *Symposium of Solid Modeling and Applications*, pp. 143–151.
[116] Michelucci D. and Foufou, S., 2006, Geometric Constraint Solving: The Witness Configuration Method, Comput.-Aided Des., **38**, pp. 284–299.
[117] Michelucci, D., and Foufou, S., 2009, "Interrogating Witnesses for Geometric Constraint Solving," *SIAM/ACM Joint Conference Geometry Physics Modeling*, pp. 343–348.
[118] Shi, Z., and Chen L., 2006, "Simplified Iterative Algorithm to Solve Geometric Constraints," J. Comput.-Aid. Design Comput. Graphics, **18**, pp. 787–792 (in Chinese).