Computer-Aided Design 44 (2012) 709–720

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Computing parameter ranges in constructive geometric constraint solving: Implementation and correctness proof

Marta Hidalgo, Robert Joan-Arinyo *

Grup d'Informàtica a l'Enginyeria, Universitat Politècnica de Catalunya, Diagonal 649, 8è, Barcelona 08028, Catalonia

ARTICLE INFO

Article history:
Received 8 November 2011
Accepted 29 February 2012

Keywords:
Parametric-based CAD
Geometric constraint solving
Parameter range computation
Configuration space

ABSTRACT

In parametric design, changing values of parameters to get different solution instances to the problem at hand is a paramount operation. One of the main issues when generating the solution instance for the actual set of parameters is that the user does not know in general which is the set of parameter values for which the parametric solution is feasible. Similarly, in constraint-based dynamic geometry, knowing the set of critical points where construction feasibility changes would allow to avoid unexpected and unwanted behaviors.

We consider parametric models in the Euclidean space with one internal degree of freedom. In this scenario, in general, the set of values of the variant parameter for which the parametric model is realizable and defines a valid shape is a set of intervals on the real line.

In this work we report on our experiments implementing the van der Meiden Approach to compute the set of parameter values that bound intervals for which the parametric object is realizable. The implementation is developed on top of a constructive, ruler-and-compass geometric constraint solver. We formalize the underlying concepts and prove that our implementation is correct, that is, the approach exactly computes all the feasible interval bounds.

1. Introduction

Many applications in computer-aided design, computer-aided manufacturing, kinematics, robotics or dynamic geometry are conveniently modeled by geometric problems defined by geometric constraints with parameters, some of them representing dimensions. These generic models allow the user to easily generate specific instances for various parameter and constraint values.

When parametric models are used in real applications, it is often found that instantiation may fail for some parameter values. Assuming that failures are not due to bugs in the system, they should be attributed to a more basic problem, that is, a certain combination of constraints in the model and values of parameters do not define a valid shape. We consider parametric models of geometric objects in the Euclidean plane with one degree of freedom corresponding to a variant parameter, after discounting rotations and translations of the entire object.

In general, the set of values of the variant parameter for which the parametric model is realizable and defines a valid shape is a set of intervals on the real line. The goal of this work is to implement the van der Meiden et al. approach [1,2], to figure out the set of values of the variant parameter that bound these intervals. The approach is built on top of a constructive ruler-and-compass solver. We prove that the algorithm is correct in the sense that it yields all and only bounds of the interval feasible values.

Computing the set of parameter values for which a parametric object is realizable is a long standing problem. However, the literature published concerning this problem is scarce. Shapiro and Vossler, [3], and Raghothama and Shapiro, [4–6], developed a theory on validity of parametric family of solids by investigating the relationship between Brep and CSG schemas in systems with dual representations for solid modeling. The formulation is built on formalisms of algebraic topology. Unfortunately, it seems a rather difficult problem transforming these formalisms into effective algorithms.

Joan-Arinyo and Mata [7] reported on a method to compute feasible ranges for parameters in geometric constraint solving under the assumption that values assigned to parameters are non-trivial-width intervals. The method applies to complex systems of geometric constraints in both 2D and 3D and has been successfully applied in the dynamic geometry field, [8]. It is a general method, the main drawback, however, is that it is based on numerical sampling.

Hoffmann and Kim [9] developed a constructive approach to calculate parameter ranges for systems of geometric constraints that include sets of isothetic line segments and distance constraints between them. Model instantiation for distance parameters within

* Corresponding author. Tel.: +34 93 401 66 69; fax: +34 93 401 60 50.
E-mail addresses: mhidalgo@lsi.upc.edu (M. Hidalgo), robert@lsi.upc.edu (R. Joan-Arinyo).

0010-4485/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2012.02.012

the ranges output by the method preserve the topology of the set of isothetic lines.

In an illuminating work, van der Meiden [1], and van der Meiden and Bronsvoort, [2], reported on a constructive method to calculate parameter ranges for systems of geometric constraints. Constraint systems are restricted to systems of distance and angle constraints on points and straight lines in 2D or 3D spaces that are well constrained and decomposable respectively into triangular and tetrahedral subproblems. The method automatically determines the allowable range for a single parameter of the system, called *variant parameter*, that an actual solution exists for any value in the range. The method consists of two steps. First a set of values for the variant parameter, called *critical* points, [10], for which some well defined subproblem feasibility changes is computed. Once sorted, critical points define a sequence of intervals and their feasibility is established by checking feasibility at some point within each interval.

Gao and Sitharam, in [11,12], described a general result concerning the computation of critical points for 2D problems with one degree of freedom which include just points and distance constraints that can be abstracted as one degree of freedom Henneberg graphs. Here we consider problems including distance and angle constraints that can be abstracted as tree decomposable graphs, a superset of Henneberg graphs.

The van der Meiden et al. method is the subject of our study. In Section 2 we recall basic concepts on constructive geometric constraint solving. Section 3 formalizes the geometric constraint problem with one variant parameter. The van der Meiden method and our implementation are described in Sections 4 and 5 respectively. Section 6 is devoted to prove that the method is correct. Section 7 describes a case study to illustrate how the approach works. Finally, in Section 8 we offer a short discussion.

2. Preliminaries

First we recall some basic concepts related to geometric constraint solving in general. Then we focus on the constructive technique. For an in depth discussion on this topic see, for example, [13–23].

2.1. Geometric constraint problems

In this paper we focus on the basic constraint problem defined as follows. Given a set of geometric elements G and a set of constraints between them C, place each geometric element in such a way that the constraints are fulfilled. We consider 2D geometric constraint problems defined by a set of geometric elements like points, lines, line segments, circles and circular arcs with fixed radius, along with a set of constraints like distance, angle, incidence and tangency between any two geometric elements.

Fig. 1(a) shows an example of geometric constraint problem consisting of six points $\{a, b, c, d, e, f\}$, nine point–point distance constraints $\{d_i, 1 \leq i \leq 9\}$. Fig. 1(b) shows the geometric constraint problem abstracted as a graph where one node represents one geometric element and one labeled edge represents a geometric constraint defined on the two geometric elements the edge connects. In what follows we will represent geometric constraint problems as graphs.

In what follows a geometric constraint problem will be denoted by a tuple $\Pi = \langle G, C, P\rangle$ where G is the set of geometric objects, C the set of constraints defined on G and P is the set of parameters of constraints in C.

The constraint solving community is mainly interested in objects which are invariant under rigid transformations of translation and rotation. This property is known as *rigidity*. The intuitive concept of rigidity, the one that will be used here, is

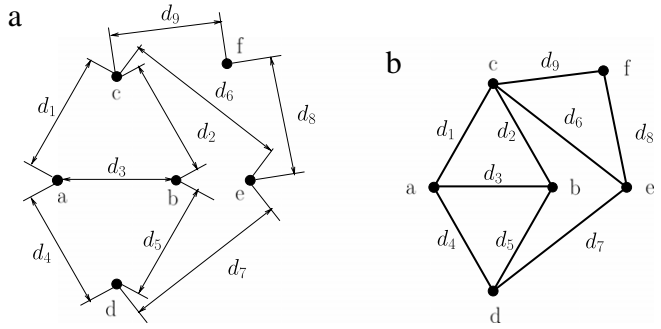

Fig. 1. Geometric constraint problem example. (a) Geometric sketch. (b) Geometric constraint problem abstracted as a graph.

defined from the number of solutions of the considered problem. In this context, geometric constraint problems are categorized in three different families:

1. Well constrained problems are geometric constraint problems with a non-empty and finite set of solutions.
2. Over-constrained problems are those problems with no actual solution. Generally, the elimination of one or more constraints results in a well constraint problem.
3. Under-constrained problems are geometric constraint problems for which an infinite set of solutions exists. In these cases, not enough constraints are given.

In this work we only consider well constrained problems.

2.2. Constructive geometric constraint problem solving

Geometric constraint solving is arguably a core technology of computer aided design and, by extension, geometric constraint solving is also applicable in virtual reality and is closely related in a technical sense to geometric theorem proving. For solution techniques, geometric constraint solving also borrows heavily from symbolic algebraic computation and matroid theory.

Many techniques have been reported in the literature that provide powerful and efficient methods for solving systems of geometric constraints. For a review, see [24]. Among all the geometric constraint solving techniques, our interest here focuses on the one known as *constructive*.

Constructive geometric constraint solving is a technique widely used in the geometric constraint solving field. We briefly recall here the main features underlying this technique. An architecture for constructive solvers is illustrated in Fig. 2 where square boxes are *functional units* and rounded boxes are *data entities*. The functional units are the analyzer, the index selector and the constructor. The data entities are the geometric constraint problem, the construction plan, the parameters assignment and the index assignment.

In this technology, the user defines a geometric constraint problem by sketching some geometric elements taken from a given repertoire (points, lines, circles, etc.) and annotates the sketch with a set of geometric relationships, called *constraints*, (point–point distance, point–line distance, angle between two lines and so on), that must be fulfilled.

Given the geometric constraint problem, $\Pi = \langle G, C, P\rangle$, the analyzer is responsible for figuring out whether the solver is able to solve the problem. If the answer is positive, the analyzer outputs the solution as a sequence of construction steps, known as *construction plan*, that will place the geometric elements in positions for which the constraints hold.

Fig. 3 shows a construction plan for the constraint problem given in Fig. 1. The meaning of each construction step is the usual. For example, *origin*() stands for the origin of an arbitrary