Using the witness method to detect rigid subsystems of geometric constraints in CAD

Dominique Michelucci, Simon Thierry, Pascal Schreck, Christoph Fünfzig, Jean-David Génevaux

Using the witness method to detect rigid subsystems of geometric constraints in CAD

Dominique Michelucci* Pascal Schreck†
Simon E.B. Thierry† Christoph Fünfzig*
Jean-David Génevaux†

September 1st – 3rd, 2010

Abstract

This paper deals with the resolution of geometric constraint systems encountered in CAD-CAM. The main results are that the witness method can be used to detect that a constraint system is over-constrained and that the computation of the maximal rigid subsystems of a system leads to a powerful decomposition method.

In a first step, we recall the theoretical framework of the witness method in geometric constraint solving and extend this method to generate a witness. We show then that it can be used to incrementally detect over-constrainedness. We give an algorithm to efficiently identify all maximal rigid parts of a geometric constraint system. We introduce the algorithm of W-decomposition to identify all rigid subsystems: it manages to decompose systems which were not decomposable by classical combinatorial methods.

Keywords: Geometric Constraints Solving, witness configuration, Jacobian matrix, rigidity theory, W-decomposition

1 Introduction

Geometric constraints solving in Computer-Aided Design (CAD) aims at yielding a figure which meets some incidence and metric requirements (*e.g.* distances between points or angles between lines), usually specified in graphical form. Formally, a geometric constraint system (GCS) consists in constraints (predicates), unknowns (geometric entities) and parameters (metric values). Solutions are returned as the coordinates of the geometric entities. The left of figure 1 shows an example of a technical sketch, and the right shows a possible solution.

The literature describes a number of different approaches to solve geometric constraint systems:

*LE2I, UMR CNRS 5158, Université de Bourgogne
†LSIIT, UMR CNRS 7005, Université de Strasbourg

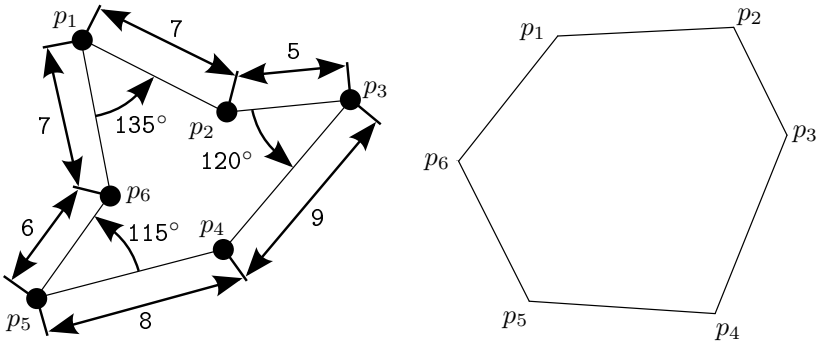

Figure 1: A 2D technical sketch (left) and a possible solution (right).

- algebraic methods consist in translating the GCS into a set of equations and working on the equation system, thus forgetting the geometrical background. Algebraic methods can be classified in numerical methods [22] (iterative computations converging to an approximate solution from initial values given by the user) and symbolic methods [2, 11] (direct computations on the equations – these methods are seldom used because of their complexity),
- geometric methods use the geometric knowledge to solve the system: graph-based methods [6, 9, 22, 28, 29, 31] compile this knowledge into algorithms which consider only combinatorial and connectivity criteria, rule-based methods [3, 17] deduce constructions plans by an explicit use of geometric rules,
- hybrid methods [4, 8, 18] alternate algebraic and geometric phases of computations to use the power of both approaches.

For more details on geometric constraint solving, see [12]. A general trend, both to reduce complexity and to enhance resolution power, is to decompose the GCS into solvable subsystems and to assemble their solutions [4, 5, 9, 13, 15, 22, 28, 29, 31, 33]. For instance, on the 2D example of figure 1, it is easy to separately solve each "triangle" ($p_1p_2p_6$, $p_2p_3p_4$ and $p_4p_5p_6$) and then assemble them. For a detailed survey of decomposition methods, see [16].

Notice that, on the example of figure 1, if one removes one of the triangles, say $p_2p_3p_4$, and then tries to solve the remaining system, one needs to add information from the solved subsystem, otherwise the remaining system becomes articulated. This piece of information is called the boundary [24]. Although several methods exist to find the relevant information in specific resolution frameworks [28], no general algorithm yet exists to compute the boundary without adding too much information.

Indeed, it is important for resolution methods, especially for graph-based methods, that the system does not have too few or too many constraints. Loosely speaking, a system is called

- under-constrained if it has an infinite number of solutions because there are not enough constraints to pin down every geometric entity,
- over-constrained if it has no solution because of constraint contradictions,
- well-constrained if it has a finite positive number of solutions.

Invariance of rigid systems by displacements is generally taken into account by anchoring a point and a direction in 2D, a point and two directions in 3D. The point and the direction are called a *reference* for the displacements. Other transformation groups may be considered [30].

A lot of work has been done about the detection of over-constrainedness [14, 27] or under-constrainedness [19, 32, 37] and more generally about the characterization of rigidity [21, 20, 30, 35]. Yet, methods described in the literature may fail to consider the consequences of mathematical theorems that are not explicitly taken into account in the construction of the resolution algorithm. Since a theorem list cannot be exhaustive, it is impossible to develop a rule-based or graph-based algorithm that detects geometric properties induced by mathematical theorems.

In this article, we extend the witness method [25] to address several problems cited above: how to determine the constrainedness level of a GCS without being tricked by mathematical theorems (see for instance figure 6); how to efficiently detect all maximal well-constrained subsystems of a given GCS; how to decompose a well-constrained system into the set of all its minimal well-constrained subsystems.

For conciseness reasons, in the rest of this paper, we consider 2D systems, unless explicitly mentionned otherwise. Yet, all algoritms can be extended to 3D systems with nearly no changes and, most of the time, the only modification to be made for the text to be valid in 3D is to exchange mentions of three degrees of freedom/parameters with mentions of six degrees of freedom/parameters.

This article is organized as follows: section 2 recalls the principles of the witness method and gives a way to generate a witness; section 3 demonstrates that an incremental version of the Gauss-Jordan elimination has the same computational cost than the original version but allows to detect overconstrainedness in all cases; section 4 gives algorithms to efficiently identify the maximal rigid subsystems of an articulated system; section 5 deduces from these algorithms a method to further decompose a rigid system into rigid subsystems; finally, section 7 concludes and gives perspectives to this work.

2 The witness method

2.1 Principle

The witness method comes from ideas of Structural Topology, or Rigidity Theory [10] where the question of rigidity is studied through the notion of frameworks. A *framework* is a triple (V, E, p) where (V, E) is a graph and $p : V \to \mathbb{R}^d$ a realization of the graph, which maps the vertices of V to points of dimension d. Thinking of graph edges as rigid bars and of vertices as articulation points, the main goal of combinatorial rigidity is to answer "Is (V, E, p) rigid?", *i.e.* it admits only rigid motions as a whole, no deformations.

Infinitesimal flexion. In Rigidity Theory, an *infinitesimal flexion* is a map $q : V \to \mathbb{R}^d$ such that $(p(i) - p(j)) \cdot (q(i) - q(j)) = 0$, for each $(i, j) \in E$. A framework is called *infinitesimally rigid*, if the only infinitesimal flexions arise from the direct isometries of $\mathbb{R}^d$, *i.e.* the translations and rotations.

Under mild assumptions concerning incidence relationships, if one framework (V, E, p_0) is infinitesimally rigid then almost all frameworks (V, E, p) are infinitesimally rigid. And the *infinitesimal rigidity* implies the *rigidity* of the framework. Note that there are counter-examples for the converse, which contain special incidences.

In other words, a framework in rigidity theory corresponds to the realization of a geometric constraint system where all constraints are point-to-point distance constraints: such a system is generically well-constrained up to direct isometries if it is generically rigid. This was generalized by Michelucci et al. [25, 26] to metric constraints over points, lines, etc. (distances and angles) and to incidence constraints (colinearities in 2D and 3D, coplanarities in 3D).

In CAD when the designer draws a sketch, he/she has a solution X_0 for a system $F(X, A_e) = 0$, with some parameter values A_e read on the sketch. Then the goal is a solution for the system $F(X, A_a) = 0$, where A_a are the values given for the dimensioning.

Witness. Let $F(X, A) = 0$ be a constraint system, where X are the unknowns and A the parameters. We suppose that $F(X, A)$ is differentiable. A *witness* is then a solution X_0 of $F(X, A) = 0$ for some parameter values A_e.

Using a Taylor expansion for a small perturbation around the solution X_0 of $F(X, A_e) = 0$, we have

$$F(X_0 + \varepsilon v, A_e) = F(X_0, A_e) + \varepsilon F'(X_0, A_e)v + O(\varepsilon^2)$$

where v can also be seen as the instant velocity of each object involved in the system and ε is a small time step. Thus, if an infinitesimally small perturbation is another solution of $F(X, A_e)$, we must have

$$F'(X_0, A_e)v = 0$$

The space of the infinitesimal motions allowed by the constraints at the witness is then given by $\ker(F'(X_0, A_e))$. Note that

- the matrix $F'(X_0, A_e)$ is known as the Jacobian of system $F(X, A_e) = 0$ taken at point X_0;

- when all constraints are point-to-point distances, the Jacobian is the rigidity matrix considered in Rigidity Theory;
- for other constraints with parameters the genericity conditions are more complicated than in the combinatorial case: a parameter value A_e and a corresponding solution X_0 are generic if the root is an implicit function of the parameters in some open neighborhood of (X_0, A_e); for instance, for a triangle specified with three length parameters, this condition forbids that one length is the sum of the others; more generally this condition implies that the matrix

$$\begin{pmatrix} \partial F(X,A)/\partial X & \partial F(X,A)/\partial A \\ 0 & Id \end{pmatrix}$$

has the same rank in an open neighborhood of (X_0, A_e) It remains that the generic parameter values are dense in the set of parameter values corresponding to a realization.

We give some examples for the formulation of generic constraints. For point, line, plane incidences, we assume that the corresponding constraints are specified explicitly without parameters. This is to avoid expressing point-point incidences by a distance constraint $(P_{1,x}-P_{2,x})^2+(P_{1,y}-P_{2,y})^2=d^2$ with distance parameter $d=0$. For a distance constraint $(P_{1,x}-P_{2,x})^2+(P_{1,y}-P_{2,y})^2=d^2$, the parameter $d=0$ is not generic, as the constraint is singular at the solution point. For an angle constraint $\text{angle}(P_1,P_2,P_3)=\theta$, *i.e.* $P_1P_2 \cdot P_3P_2 = l_{P_1P_2}l_{P_3P_2}\cos\theta$, the parameter values $\theta=\pm\pi$, $\theta=\pm\pi/2$, and $\theta=0$ are not generic. Similarly, point-line, line-plane incidences and line-line, plane-plane parallelism/orthogonality constraints are not expressed by angle constraints because it would introduce non-generic angles.

Typicality. A witness is *typical* if it is representative for the searched solution, *i.e.* it has the same combinatorial properties (coincidences, collinearities, coplanarities, etc.). So a random solution (X_0, A_e), $\{(X,A) : F(X,A)=0\}$ with the specified combinatorial properties is typical with probability 1 for a set of witness solutions. Note that systems exist with witness solutions, which are different in combinatorial properties, and no continuous deformation exists to transform one into the other. For an example of such a system see figure 14 in [16].

We can then study the degrees of freedom of the system by studying the rank of the Jacobian $F'(X_0, A_e)$ on a typical witness X_0, and in the case of under-constrainedness, the structure of the allowed infinitesimal motions can be deduced from the study of the kernel of $F'(X_0, A_e)$.

In the rest of this paper, we consider that rows of the Jacobian matrix represent constraints and columns represent coordinates of the unknowns. We classically denote by m the number of rows and by n the number of columns of the matrix.

2.2 Generation of a witness

The sketch is usually a witness but due to implied incidences this may not be the case. In this case, we solve the under-determined system $\{(X, A) : F(X, A) = 0\}$ for a witness (X_0, A_e). In the subdivision solver presented in [7], the nonlinear monomials x_i^2 and $x_i x_j$ for $i < j$ are replaced by additional variables $x_{i,i}$ and $x_{i,j}$, which are enclosed in a polytope $B_D(x_i, x_{i,i}, x_{i,j,i<j}) \geq 0$ with halfspaces given by the non-negativity of relevant Bernstein polynomials (*Bernstein polytope*). The quadratic constraint system becomes a polytope $S(x_i, x_{i,i}, x_{i,j,i<j}) \geq 0$ after rewriting into the additional variables $x_{i,i}$ and $x_{i,j}$. The subscript D of $B_D(x_i, x_{i,i}, x_{i,j,i<j}) \geq 0$ indicates that this polytope depends on the domain D. In this way, bounds for the solution domain of quadratic polynomials can be expressed as two linear programs

$$\begin{array}{l} \min x_i \text{ and } \max x_i \\ S(x_i, x_{i,i}, x_{i,j,i<j}) \geq 0 \\ B_D(x_i, x_{i,i}, x_{i,j,i<j}) \geq 0 \end{array}$$

Domain bounds are computed by linear programming in order to reduce the current solution domain D. If the feasible set is empty, which is detected by linear programming, then the current domain box contains no solution. Otherwise, we can perform a sequence of reductions and bisections of domain boxes until the domain box $D = [\underline{x_1}, \overline{x_1}] \times \ldots \times [\underline{x_n}, \overline{x_n}]$ is δ-small: $(\overline{x_i} - \underline{x_i}) < \delta$ for all i. These δ-small boxes cover the solution set piecewise.

The subdivision solver requires a domain box to start the search. The intervals for generic parameter values of constraints are easy to find: angle parameters $\cos\theta$ ($\cos\theta$ instead of θ to avoid trigonometric functions in the solver) are in $[-1 + \epsilon, -\epsilon]$ or $[\epsilon, 1 - \epsilon]$ with a small, arbitrary ϵ; intervals for distance parameters d can be obtained from magnitude bounds of the point coordinates. Finding a bound on the magnitude of any root [36], would be necessary to prove that the system has no solution. For the problems here, a bound on the point coordinates is known beforehand.

In order to enumerate all solutions of a system, we used mid-bisection of the largest interval in [7], which minimizes the height of the exploration tree while cycling through dimensions. For the case of determining a single solution as fast as possible, the choice of the smallest interval (greater or equal δ) is beneficial as setting variables to values allowing solutions improves the effectiveness of the domain reduction step.

We select the next domain box (of smallest minimum side length greater than δ) for reduction and bisection at random. In this way, we find a solution box containing a random solution, and we take the box center projected onto the solution set as a witness.

As examples, we show two systems of different difficulty. In figure 2, two triangles with a common point p_0 are specified by six side lengths. In the random solution, the side lengths are all different. In figure 3, four points and five lines with 10 point-line incidences are specified by four angle parameters and a distance parameter. The left part shows a solution with symmetric and nice

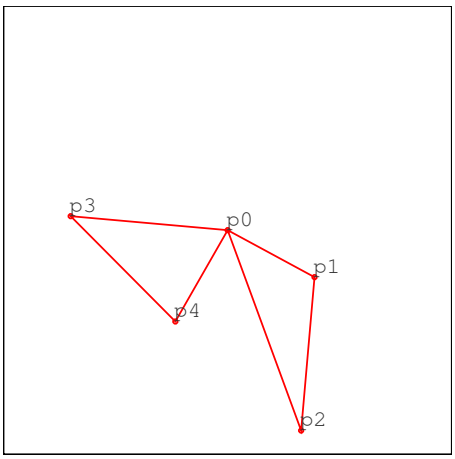

Figure 2: "The butterfly": 2D system with 5 points and 6 distance parameters $d(p_0, p_1)$, $d(p_1, p_2)$, $d(p_2, p_0)$, $d(p_0, p_3)$, $d(p_3, p_4)$, $d(p_4, p_0)$.

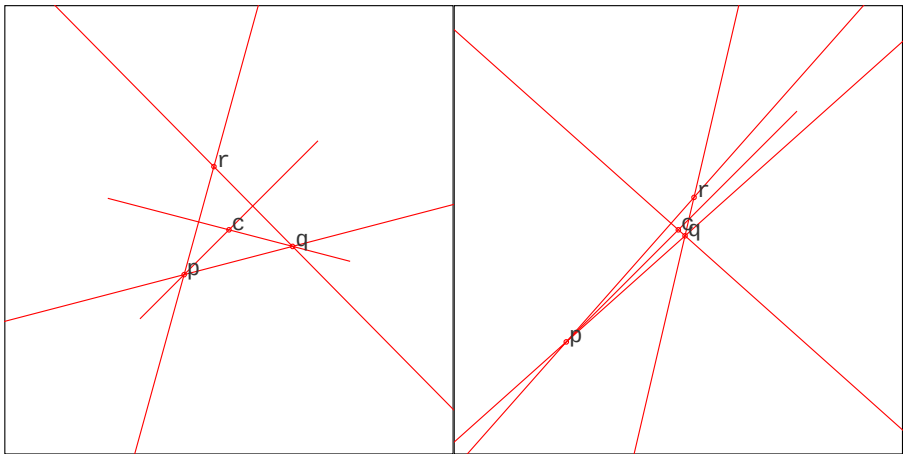

Figure 3: 2D system of 4 points and 5 lines with 10 point-line incidences, 4 angle parameter angle(qp, cp), angle(cp, rp), angle(rq, cq), angle(cq, pq) and 1 distance parameter $d(r, c)$. Symmetric solution (left) and random, typical witness solution (right).

shaped triangles, obtained by additional minimum distance constraints between the triangle points. In the right part, a typical witness solution is shown, which was found at random. It is used for further analysis.

3 Over-constrainedness

We already showed in section 1 that the detection of over-constrainedness is a complicated yet essential problem in the field of geometric constraints solving.

In this section, we show that the use of the witness method leads to an efficient and robust detection of redundancy in geometric constraints.

We also show the usefulness of the witness method to enhance robustness of decomposition methods by an accurate computation of the boundary.

3.1 Incremental detection of redundancy

We showed in [25] that it is possible to interrogate a witness in order to detect whether a set of constraints is dependent or not. Indeed, it is possible to compute the rank of the Jacobian matrix at the witness and to compare it with the number of constraints. However, finding a maximal independent subset of a dependent set is not a trivial problem. Working on the witness, the naive idea would be to try and remove constraints one by one and, at each step, compute the rank again to determine if the constraint is redundant with the remaining set. If the rank of $\mathcal{S} - c$ equals the rank of $\mathcal{S}$, then constraint c is redundant and can be removed. Performed this way, the removal of redundant constraints is expensive. Yet, considering an incremental construction of the geometric constraint system allows to identify the set of redundant constraints with no additional costs in comparison to the basic detection of redundancy.

Indeed, consider a geometric constraint system $\mathcal{S}$ with no redundancy between the constraints. Applying the Gauss-Jordan elimination method on the Jacobian matrix at the witness leads to a matrix $J' = (IP)$ with I a $m \times m$ diagonal matrix and P a $m \times f$ matrix, $f = n - m$ being the number of actual degrees of freedom of the system. This method has a known complexity of $\mathcal{O}(\min(n, m)nm)$. Let us now consider a system $\mathcal{S}'$ with $\mathcal{S} \subset \mathcal{S}'$. In order to know if $\mathcal{S}'$ is over-constrained, one only needs to incrementally add the geometric entities and the constraints (bearing in mind that a constraint can be inserted only when the geometric entities it concerns are all in the system) of $\mathcal{S}' - \mathcal{S}$ to $\mathcal{S}$ and applying Gauss-Jordan again. Since the leftmost part of the matrix is the diagonal, the number of operations is at most $2\min(m, n)f$: for each row of I, each non-zero element of P must be multiplied and added to the new row. The number of operations is in fact far smaller, since the number of zero elements in the new row of the matrix is high.

Proceeding incrementally does not raise the number of operations: it only changes the order of the operations. Indeed, the classical Gauss-Jordan elimination method consists in column-by-column operations: for each column c, divide row c by $J_{c,c}$, then substract $J_{r,c}$ times this new row from row r for every r, so

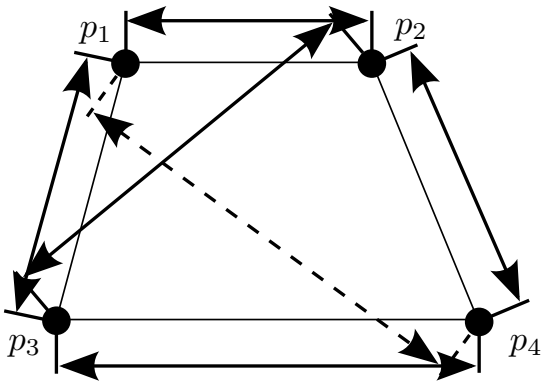

Figure 4: "The kite": over-constrained 2D system with 4 points and 6 distances. Without the dotted constraint, the system is rigid.

Table 1: The Jacobian matrix of table 2 at a witness. The Gauss-Jordan elimination method was used on the first five rows. The sixth row is redundant ($r_6 = r'_2 - r'_1$)

	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
r'_1	1	0	0	0	0	$-\frac{4}{5}$	-1	$\frac{4}{5}$
r'_2	0	1	0	0	0	$-\frac{4}{5}$	0	$-\frac{1}{5}$
r'_3	0	0	1	0	0	$-\frac{3}{5}$	-1	$\frac{3}{5}$
r'_4	0	0	0	1	0	$-\frac{1}{5}$	0	$-\frac{4}{5}$
r'_5	0	0	0	0	1	$\frac{2}{5}$	-1	$-\frac{2}{5}$
r_6	-1	1	0	0	0	0	1	-1

that column c is a null vector except for the c-th value. With the incremental calculus of the reduced row echelon form, one proceeds row by row: for each row r, substract $J_{c,c}$ times row c for each $c < r$, then divide row r by $J_{r,r}$ so that the $r-1$ first elements of row r are zero and the r-th element is 1. Thus, the overall complexity of the incremental computation of the reduced row echelon form of J is also of $\mathcal{O}(\min(n,m)nm)$.

The incremental version of the Gauss-Jordan elimination has the same complexity as the one-step version, but has a major advantage in our case: at each step, when a constraint is inserted, one may compare the new rank with the previous one and thus detect a redundant constraint. With exactly the same number of operations as in the case of the classical Gauss-Jordan elimination, one obtains the reduced row echelon form of the Jacobian matrix together with the list of redundant constraints.

Let us consider the 2D example of figure 4. The Jacobian matrix of this system is shown in table 2. Consider the following witness: $p_1 = (2,7)$, $p_2 = (5,6)$, $p_3 = (1,1)$ and $p_4 = (6,3)$. The Jacobian at this witness is shown in table 1, with a partial Gauss-Jordan elimination, since the sixth row has not

Table 2: The Jacobian matrix for the system of figure 4.

	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
r_1: dist(p_1, p_2)	$x_1 - x_2$	$y_1 - y_2$	$x_2 - x_1$	$y_2 - y_1$	0	0	0	0
r_2: dist(p_1, p_3)	$x_1 - x_3$	$y_1 - y_3$	0	0	$x_3 - x_1$	$y_3 - y_1$	0	0
r_3: dist(p_2, p_4)	0	0	$x_2 - x_4$	$y_2 - y_4$	0	0	$x_4 - x_2$	$y_4 - y_2$
r_4: dist(p_3, p_4)	0	0	0	0	$x_3 - x_4$	$y_3 - y_4$	$x_4 - x_3$	$y_4 - y_3$
r_5: dist(p_2, p_3)	0	0	$x_2 - x_3$	$y_2 - y_3$	$x_3 - x_2$	$y_3 - y_2$	0	0
r_6: dist(p_1, p_4)	$x_1 - x_4$	$y_1 - y_4$	0	0	0	0	$x_4 - x_1$	$y_4 - y_1$

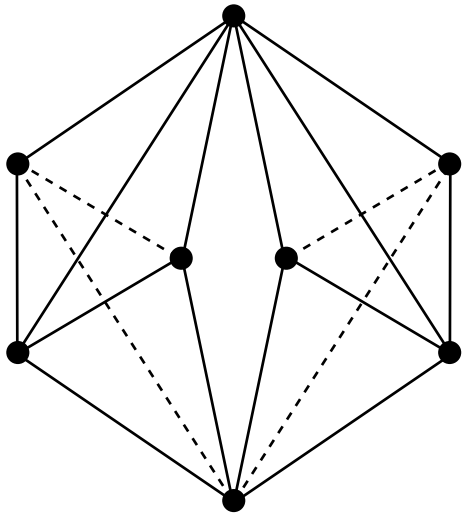

Figure 5: "The double-banana": famous counter-example to the extension of Laman's characterization of rigidity in 3D. Each segment represents a distance constraint.

been modified. That is, table 1 shows the matrix obtained by performing the incremental version of the Gauss-Jordan elimination, after inserting the sixth constraint but before performing Gauss pivoting on it. It is easy to see that the sixth row is redundant, since it can be obtained by substracting the first row from the second one. Thus, we detected the over-constrainedness.

For a more complex and famous example, consider the double-banana (see figure 5): adding the last constraint of the double-banana leads to a zero-filled row in the Jacobian matrix at the witness. If one considers an example with higher connectivity [23], our method still succeeds to efficiently detect over-constrainedness.

Moreover, the witness method correctly handles redundancy in under-constrained cases, where graph-based methods are helpless because they do not consider geometric theorems. For instance, consider the 2D example of figure 6. It is unlikely that a graph-based method can ever detect the fact that point y is fixed,

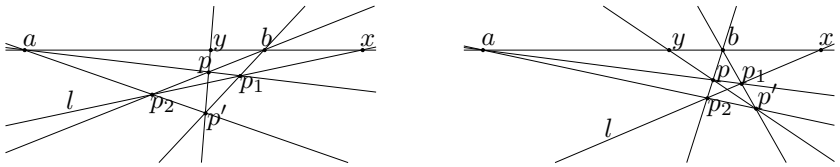

Figure 6: In 2D, given three aligned points a, b and x and for any point p and line l traversing x, y is unchanged: $p_1 = (ap) \cap l$, $p_2 = (bp) \cap l$, $p' = (ap_2) \cap (bp_1)$, $y = (ab) \cap (pp')$.

no matter what coordinates are given to point p and line l. Hence, a graph-based method would see this system as a system with 8 remaining degrees of freedom (5 for the three aligned points a, b and x, 1 for line l traversing x and 2 for point p) and would consider that adding a constraint distance between points a and y removes a degree of freedom. The witness method, however, detects that this new distance constraint is redundant and that the unknown y is determined by the system though l and p can be chosen at random.

3.2 Computation of well-constrained boundary systems

This easy and efficient way to compute a maximal independent subset of the constraints is also useful in decomposition to make sure that the boundary of a subsystem is not over-constrained.

Recall that the boundary of a system $\mathcal{S}'$ according to a system $\mathcal{S}$ is the set of all information computable in $\mathcal{S}'$ about geometric entities which are both in $\mathcal{S}$ and $\mathcal{S}'$. For instance, if $\mathcal{S}'$ is a rigid system which shares three points with system $\mathcal{S}$, then the boundary of $\mathcal{S}'$ contains the following displacement-invariant constraints:

- the three point-point distances,
- the three angles between the sides of the triangle.

It is easy to see that if the boundary of a subsystem is not added after removal of the subsystem from a rigid GCS, then the remaining GCS becomes under-constrained because information is lost. For instance, consider the GCS of figure 4 without the constraint shown with dotted lines. The triangle $p_1p_2p_3$ is rigid and trivially solved. If it is removed from the system, the remaining GCS is a 2-bars system containing two distance constraints: $p_3 - p_4$ and $p_2 - p_4$. This remaining system has solutions which are not subfigures of the global GCS, since the angle between both bars may vary.

To get rid of this problem, one may add the boundary of the solved subsystem to the remaining system [24]. In the example above, the boundary of triangle $p_1p_2p_3$ consists of the distance between points p_2 and p_3. With a bigger boundary, a new problem arises. Consider, for instance, a rigid subsystem which shares three points with the remaining system. One can compute the values of the three point-point distances, but also the values of the three angles. Thus,

the boundary is an over-constrained GCS with three points and six constraints. Although, formally, the system is not over-constrained since the metrics are consistent, it is structurally over-constrained, which means that any combinatorial method will fail to continue the solving process.

Using our incremental Gauss-Jordan elimination method, one can compute a well-constrained subset of the boundary system which contains all the information to generate the rest of the boundary system. One adds all the constraints of the boundary one by one to an empty system. If the last inserted constraint is redundant with the previous ones, one removes it.

Note that all maximal independent subsets of the constraints are geometrically equivalent, *i.e.* the computed boundary will depend on the order in which constraints are considered, but whatever this order is, the result will be correct.

4 Detection of maximal rigid subsystems in articulated systems

In this section, we show how the witness method can be used to efficiently detect all maximal rigid subsystems (MRS) of a geometric constraint system, even with systems for which graph-based methods would fail to detect rigidity. We give a basic algorithm based on a series of Gauss-Jordan eliminations then show two ways to enhance computation speed.

The basic idea of our MRS detection algorithm is to study which geometric entities are fixed when one anchors a reference for the displacements (see [24] or [30] for a formal definition of references). In the witness framework, anchoring a reference for the displacements consists in switching columns in the Jacobian matrix so as to put the three columns of the reference in the right-most positions. Indeed, performing a Gauss-Jordan elimination diagonalizes the matrix from the left and thus consists in expressing the different coordinates as functions of the right-most columns (the ones that do not belong to the identity part of the matrix). For instance, table 1 shows the reduced row echelon form of the Jacobian matrix at the witness for the GCS of figure 4. Since this GCS is rigid (with the redundant constraint removed), three columns do not belong to the identity part of the matrix: they correspond to coordinates x_4, y_4 and y_3, which form a reference for the system. All other coordinates can be expressed in function of these three coordinates. For instance, the first line of the matrix must be interpreted as $x_1 - \frac{4}{5}y_3 - x_4 + \frac{4}{5}y_4 = 0$, *i.e.* $x_1 = \frac{4}{5}y_3 + x_4 - \frac{4}{5}y_4$.

When the GCS is not rigid, three parameters are not enough to anchor all entities. There are then more than three columns at the right of the identity. Table 3 shows the reduced row echelon form of the Jacobian matrix at a witness for the GCS of figure 7. Notice that columns y_2 and y_4 were moved to the right, since it would have been impossible to find a pivot and finish the Gauss-Jordan elimination otherwise. All coordinates can be expressed as functions of y_2, y_4, y_6, x_7 and y_7. Indeed, a reference for this GCS can consist in point p_7, direction p_7-p_6, direction p_5-p_4 and direction p_3-p_2.

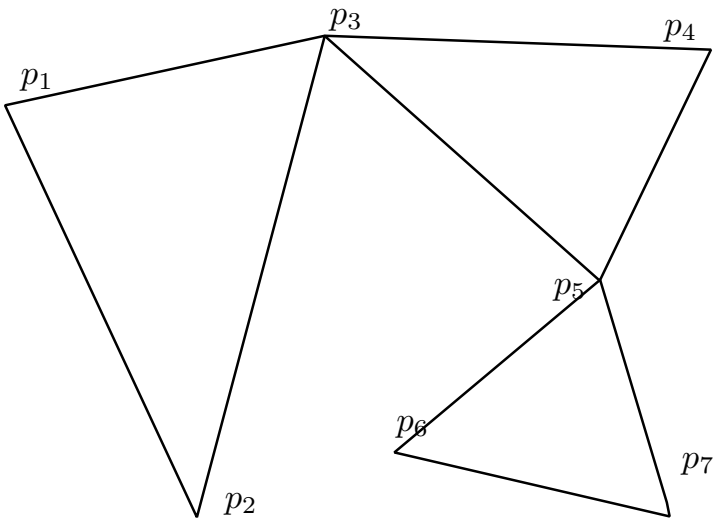

Figure 7: 2D articulated chain made of three rigid triangles. Distance constraints are implicitly represented by the segments.

An important result to identify MRSs comes from the zeros in columns y_2 and y_4. Rows 7, 8 and 9 of table 3 can be interpreted as the fact that the values of x_5, y_5 and x_6 depend only on those of y_6, x_7 and y_7. Put differently, if one anchors a reference for the displacements by pinning down p_7 and direction p_7-p_6, then points p_6 and p_5 are fixed, *i.e.* $p_5p_6p_7$ is a rigid subsystem.

A naïve algorithm immediately arises, based on anchoring a reference for the displacements, *i.e.* switching columns to have the corresponding columns on the right of the Jacobian matrix and identifying the parts of the GCS which are fixed. The pseudo-code is shown as algorithm 1. In this algorithm, anchoring a reference for the displacements means switching columns so as to have the columns corresponding to the reference at the right of the Jacobian matrix. In order to not identify the same MRS twice, we anchor references only on untagged parts of the GCS, that means that at least one of the columns cannot be tagged.

Algorithm 1 Naïve MRS identification algorithm

1: $i \leftarrow 0$
2: **repeat**
3: anchor a reference for the displacements on an untagged part of the GCS
4: perform a Gauss-Jordan elimination
5: tag with label i the columns of the GCS which correspond to coordinates depending only on the reference
6: $i \leftarrow i + 1$
7: **until** all the columns are tagged

Table 3: Reduced row echelon form of the Jacobian matrix at a witness for the GCS of figure 7

	x_1	y_1	x_2	x_3	y_3	x_4	x_5	y_5	x_6	y_2	y_4	y_6	x_7	y_7
r'_1	1	0	0	0	0	0	0	0	0	$\frac{4}{3}$	$\frac{101}{18}$	$-\frac{181}{108}$	-1	$-\frac{473}{108}$
r'_2	0	1	0	0	0	0	0	0	0	$-\frac{7}{3}$	$-\frac{40}{9}$	$\frac{28}{27}$	0	$\frac{140}{27}$
r'_3	0	0	1	0	0	0	0	0	0	4	$\frac{29}{2}$	$-\frac{15}{4}$	-1	$-\frac{59}{4}$
r'_4	0	0	0	1	0	0	0	0	0	0	$\frac{9}{2}$	$-\frac{17}{12}$	-1	$-\frac{37}{12}$
r'_5	0	0	0	0	1	0	0	0	0	0	$\frac{5}{2}$	$-\frac{7}{12}$	0	$-\frac{35}{12}$
r'_6	0	0	0	0	0	1	0	0	0	0	3	$-\frac{7}{6}$	-1	$-\frac{11}{6}$
r'_7	0	0	0	0	0	0	1	0	0	0	0	$-\frac{2}{3}$	-1	$\frac{2}{3}$
r'_8	0	0	0	0	0	0	0	1	0	0	0	$-\frac{1}{6}$	0	$-\frac{5}{6}$
r'_9	0	0	0	0	0	0	0	0	1	0	0	$-\frac{7}{6}$	-1	$\frac{7}{6}$

The cost of this algorithm depends on the number k of MRSs: for each of them, it performs a Gauss-Jordan elimination only once, so that the total cost is $\mathcal{O}(k \min(n, m)nm)$.

This cost can be reduced to $\mathcal{O}((k+\min(n, m))nm)$ by not starting the Gauss-Jordan elimination from scratch for each MRS. At the end of line 6 in the algorithm, the Jacobian matrix at the witness is in reduced row echelon form. By switching the columns in an appropriate way, one needs only perform the Gauss-Jordan pivot operation on two to three columns. Indeed, by looking at the constraint graph, it is possible to select a new reference for the GCS (*i.e.* a set of f columns, f being the number of degrees of freedom of the GCS) which satisfies the following conditions:

- it includes a reference for the displacements which is not totally tagged,
- each identified MRS is fixed, *i.e.*
  - the reference includes three coordinates in the MRS,
  - the MRS shares a geometric entity with a fixed MRS and the reference includes a coordinate in the MRS.

To select this reference, one only needs to consider a geometric entity which is in an already identified MRS and which is linked by a constraint to an untagged entity. More cases occur with systems for which the constraint graph has several connected components or with systems with implicit points (*e.g.* similarity-invariant systems with only lines and angles), but the principle remains the same. Thus, in most cases, one only needs to switch two columns, so as to change the point in the reference. Three switches happen with disconnected graphs. Algorithm 2 shows how to perform MRS identification. For the sake of simplicity, the algorithm is described for articulated GCS made of several MRSs connected by points, but it is easily extended to systems with other kinds of geometric entities.

In the case of open chains, *i.e.* GCS where all cycles in the constraint graph are included in rigid subsystems, an even less costly algorithm exists, by

Algorithm 2 MRS identification algorithm for an articulated system

1: anchor a reference for the displacements and identify and tag a first MRS
2: **repeat**
3: select a tagged point linked by a constraint to an untagged element
4: switch the columns of this point with the columns of the point in the last reference
5: perform Gauss-Jordan elimination on the two latter in order to identify a new MRS
6: tag the new MRS
7: **until** all the columns are tagged

using both the constraint graph and the Jacobian matrix. After performing the Gauss-Jordan elimination, a first MRS is identified by considering all the coordinates which depend only on the reference. From there, one can consider all the coordinates which depend on the reference and on one additional parameter. In the matrix of table 3, with the additional parameter y_4, x_3, y_3 and x_4 are fixed. Taking a look at the constraint graph, we notice that the previously identified MRS ($p_5p_6p_7$) shares only one point with the rest of the system and thus cannot "transfer" more than two degrees of displacement.

This enables us to remove the MRS and exchange the three parameters y_6, x_7 and y_7 with parameters x_5 and y_5 in the Jacobian matrix. The numerical values are not important in this process: we consider that all the values of both columns are non-zero. With this new matrix, one notices that parameters x_5, y_5 and y_4 form a reference for the displacements and that by anchoring this reference, x_3, y_3 and x_4 are fixed, *i.e.* $p_3p_4p_5$ is a rigid system. We continue this algorithm by noticing that this system shares only one point with the rest of the system, removing it and replacing it with non-zero-filled columns x_3 and y_3 and thus identifying the last MRS $p_1p_2p_3$.

When the last identified MRS shares more than one point with the rest of the system, two cases occur: either the removal of the MRS leads to two disconnected graphs (*i.e.* the MRS is in the middle of the articulated system) and one thus continues the algorithm separately on each part of the graph; or the MRS belongs to a non-rigid closed chain.

When one uses this algorithm on a GCS containing non-rigid closed chains, it leads to cases where one cannot detect the MRSs of the closed chains, because of the inter-dependance of the rigid subsystems of the chain. After identifying the first MRS of the closed chain, the algorithm is stuck because it is not possible to identify another system which depends only on three parameters. In this case, we get back to algorithm 2 to identify the different MRSs of the closed chain.

Notice that this section is about identification of maximal rigid subsystems but that since it is based on the anchoring of references, one may adapt the algorithms to identify maximal subsystems well-constrained *modulo* other transformation groups than the displacements.

5 W-decomposition of a rigid GCS

The previous section gives algorithms to identify all MRSs of a GCS. Having such an algorithm leads to a natural method to decompose a rigid geometric constraint system. We call this method W-decomposition and a system which can be decomposed by this method is said to be W-decomposable. In this section, we explain the principles of W-decomposition and give examples.

Algorithm 2 identifies maximal rigid subsystems, *i.e.* if a MRS can be decomposed in several rigid subsystems, this will not be detected. The basic idea of W-decomposition is to remove constraints from the constraint graph and see if it breaks the MRS in non-trivial MRSs, *i.e.* MRSs which are not limited to their boundary (*e.g.* a system limited to a point-point distance). If it does, then we use W-decomposition on each non-trivial MRS. Algorithm 3 gives the pseudo-code of the algorithm.

Algorithm 3 W-decomposition

Input: a rigid GCS $\mathcal{S}$ with
its constraint graph $G = (V, E)$ and
a witness W of $\mathcal{S}$
Output: a list of rigid subsystems

```
1: repeat
2:     Select a constraint e
3:     Identify MRSs of (V, E/{e}) with alg. 2
4:     while each MRS is equivalent to its boundary do
5:         Choose another constraint e and identify MRSs of (V, E/{e})
6: until all constraints are tested or there is a MRS which is not equivalent to
   its boundary
7: if no MRS bigger than its boundary is found then
8:     return list [G] //G is W-indecomposable
9: else
10:    remove all the constraints included in non-trivial MRSs
11:    insert the boundary of all non-trivial MRSs in the system//cf. section 3.2
12:    reintroduce constraint e in the system//this gives a rigid constraint system
13:    recursively W-decompose the resulting system
14:    recursively W-decompose all previously identified MRSs
15:    return the concatenation of the lists obtained in the last two lines
```

Let us illustrate this algorithm on the example of figure 8a, which represents the constraint graph of a rigid GCS. The graph is 3-connected and has two $K_{3,3}$ subgraphs, connected by three "middle" edges. Algorithm 2 detects the rigidity of the whole system. Let us consider the removal of two constraints at line 2 of algorithm 3: dotted edges e_1 and e_2.

If we remove edge e_1, the use of algorithm 2 at line 3 identifies four MRSs: the rigid $K_{3,3}$ subsystems, and each edge between them. The latter are equivalent to their boundary. Replacing the rigid hexagons by their boundaries and reintroducing edge e_1 leads to the graph of figure 8b (note that edge e_1 must be

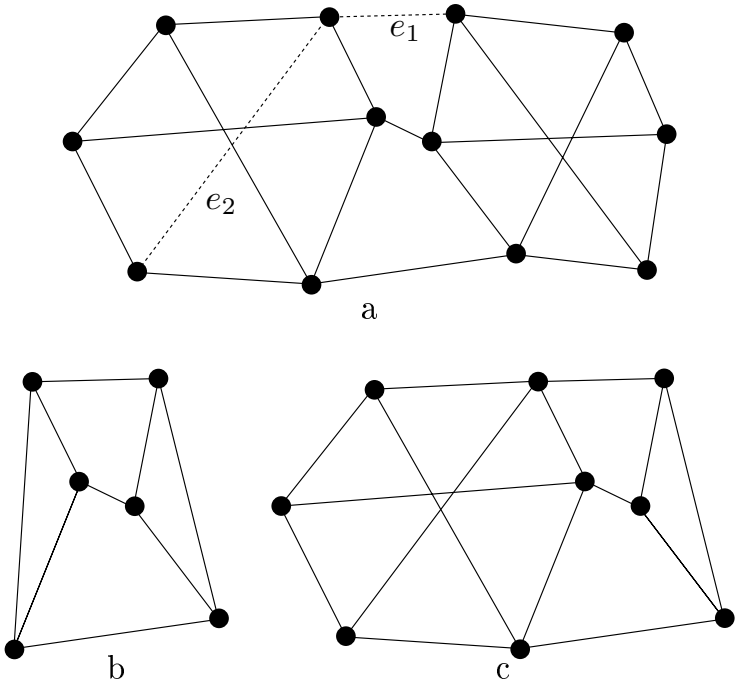

Figure 8: 2D systems where edges represent point-point distances; a: 3-connected constraint graph made of two $K_{3,3}$ graphs connected with 3 constraints; b and c: graphs obtained by replacing MRSs identified by algorithm 3 by their boundary with respectively edges e_1 and e_2 removed.

taken into account for the computation of the boundaries). The recursive use of W-decomposition (line 14) on each non-trivial MRS leads to the knowledge that they are not W-decomposable, as does the recursive use on the system of figure 8b (line 13).

If we do not remove edge e_1 but e_2 instead, the left $K_{3,3}$ subsystem of figure 8a is no longer rigid. The identification of non-trivial MRS thus only identifies the hexagon on the right of figure 8a. Once it is replaced by its boundary, we obtain the system shown on figure 8c. The recursive use of W-decomposition will then lead, after removal of one of the three "middle" edges, to the identification of the second rigid hexagon and thus to the system shown on figure 8b.

Execution time depends on the choice of the removed constraint. In the worst case, all constraints are tested: n times the algorithm 2 is used, thus the complexity is $\mathcal{O}(n^3 m)$.

Our algorithm is more powerful than algorithms found in the literature, for several reasons:

- first of all, it is independent of the connectivity of the constraint graph.

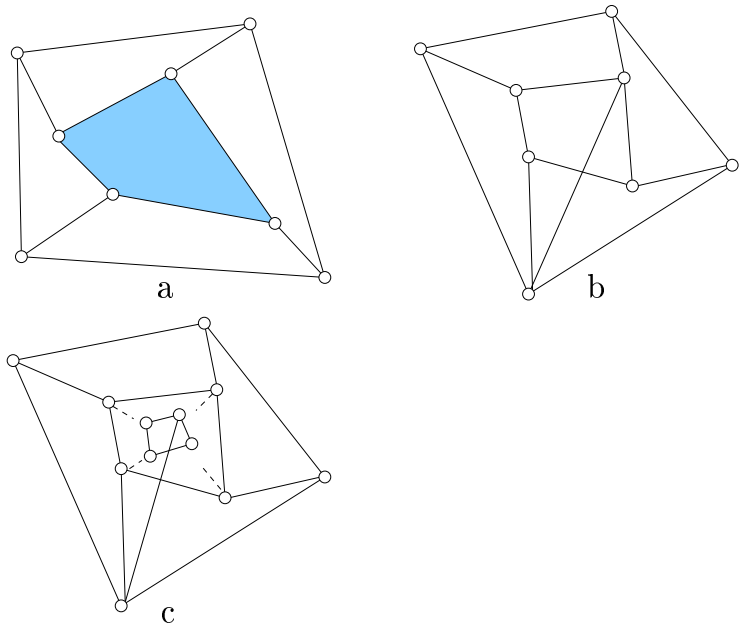

Figure 9: 2D examples for the W-decomposition: each vertex is a point and each edge represents a distance constraint. a: W-decomposable 4-connected GCS (the blue subsystem is rigid); b: W-indecomposable system; c: there are W-indecomposable systems with an arbitrary number of points.

For instance, figure 9a gives an example of a 4-connected constraint graph which is W-decomposable, no matter what is inside the inner blue part as long as it is rigid,

- second, it is also not based on a cluster formation. Since the graph of figure 9b is not decomposable by current graph decomposition methods, the system of figure 9a, with the inner part replaced by figure 9b, will also lead to a decomposition failure for these methods, whereas it is W-decomposable.

Ultimate decomposition consists in yielding a triangular equation system. For algebraic systems, Wu-Ritt decomposition or Gröbner basis with lexical order lead to such decompositions, but unfortunately, they are untractable in the CAD domain. On the other hand, W-decomposition is not as powerful as these algebraic methods since it is possible to construct an infinite family of W-indecomposable constraint systems like the one depicted in figure 9c: there is no constraint in this system such that its removal produces a system with a MRS bigger than a point-point distance. But, on the positive side, it is easy to see that

- all Owen-decomposable systems are W-decomposable (that is, articulation

pairs are detected by the choice of the deleted constraint)

- all constraint systems which are decomposable by cluster formation methods or on the search of minimal rigid parts, are also W-decomposable.

We think that the ratio of efficiency to power of decomposition is good enough to give good results in CAD even in the 3D case.

6 Robustness issue

Our method assumes it is possible to compute the rank of a set of vectors, given by their coordinates. It is a basic problem in computerized linear algebra with well-known methods. Only at first glance, it looks like an easy problem.

Since the rank is not a continuous function, it is not computable in the sense of Computable Analysis [34]. In short, Computable Analysis uses interval arithmetic with interval bounds represented using a long float arithmetic. However, the interval width is never zero. In this arithmetic, it is impossible to detect that a number (a Gauss pivot, or a determinant) is zero. On the contrary, it is possible to detect that a number is non-zero: compute a sufficiently precise interval, not containing zero.

If a rational witness is available, an exact rational arithmetic can be used. The rank of rational vectors is computable, and this apprach is practical. It is explored in [25] with a number of examples. If a rational witness is not available, like for a regular pentagon, one may theoretically resort to an exact algebraic arithmetic, for instance an algebraic arithmetic based on gap theorems [1]. Unfortunately, the large time complexity of this method makes it impractical for general systems.

We use rational arithmetic when rational witnesses are available. When no rational witness is available, and the solver is used, it provides interval approximations of witnesses. We use an *epsilon-heuristic* like the dynamic geometry softwares (Cabri Géomètre, Cinderella, GeoGebra, etc.): we decide by an epsilon threshold in the Gauss-Jordan algorithm whether vectors are dependent or not. Because all our applications of the Gauss-Jordan elimination algorithm do not depend on a special ordering of constraint rows (section 3), we can use all pivoting techniques available for it.

In practice, all geometric constraint systems met in CAD / CAM seem to have a rational witness. Systems without rational witnesses exist like for example a regular pentagon but they appear to us as artificial instances.

7 Conclusion

After proposing a way to generate a witness, we showed in this paper how the witness method could be used to detect over-constrained systems without any additional computational cost by an incremental Gauss-Jordan elimination of

the Jacobian matrix at the witness. This allows the computation of a well-constrained boundary inside the decomposition method.

We gave algorithms to identify all maximal well-constrained subsystems of a GCS, *i.e.* the system itself if it is well-constrained, or its rigid parts if it is articulated. From this algorithm, we deduced a method, called W-decomposition, to decompose a rigid GCS into the set of all its non-trivial rigid subsystems, based on the removal of a constraint and the computation of the new maximal rigid subsystems.

The method to detect over-constrainedness is efficient (the computation of the reduced row echelon form of the Jacobian matrix is performed in $\mathcal{O}(\min(n,m)nm)$) and is not tricked by mathematical theorems, even when these theorems are unknown to the developer. The MRS identification is also efficient ($\mathcal{O}(n^2m)$ with algorithm 2) and works as well with other transformation groups than the displacements. W-decomposition is performed in $\mathcal{O}(n^3m)$ in the worst case.

For conciseness reasons, the algorithms we described work on 2D systems, but they can be easily extended to 3D systems. Complexity of the algorithms is independent of the dimension.

Further research needs to be done in order to find heuristics for the optimization of W-decomposition. The example of figure 8 shows that some edges are better than others for the removal (line 2 of algorithm 3). We think that a promising track is the computation of a minimum chain covering and the search for constraints which appear in only a few chains.

References

[1] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, 1988.

[2] S.-C. Chou and X.-S. Gao. Ritt-wu's decomposition algorithm and geometry theorem proving. In *CADE '90: Proceedings of the 10th International Conference on Automated Deduction*, pages 207–220, Kaiserslautern, Germany, 1990. Springer.

[3] J.-F. Dufourd, P. Mathis, and P. Schreck. Geometric construction by assembling solved subfigures. *Artificial Intelligence*, 99(1):73–119, 1998.

[4] A. Fabre and P. Schreck. Combining symbolic and numerical solvers to simplify indecomposable systems solving. In *SAC '08: Proceedings of the 23rd ACM Symposium on Applied Computing*, pages 1838–1842, Fortaleza, Brazil, 2008. ACM.

[5] S. Foufou, D. Michelucci, and J.-P. Jurzak. Numerical decomposition of geometric constraints. In *SPM '05: Proceedings of the 10th ACM Symposium on Solid and physical modeling*, pages 143–151, Cambridge, Massachusetts, USA, 2005. ACM.

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

[7] C. Fuenfzig, D. Michelucci, and S. Foufou. Nonlinear systems solver in floating point arithmetic using LP reduction. In *SPM '09: Proceedings of the SIAM/ACM joint conference on Geometric and Physical Modeling*, pages 123–134, San Francisco, California, USA, 2009.

[8] X.-S. Gao, C. M. Hoffmann, and W.-Q. Yang. Solving spatial basic geometric constraint configurations with locus intersection. *Computer-Aided Design*, 36(2):111–122, 2004.

[9] X.-S. Gao, Q. Lin, and G.-F. Zhang. A C-tree decomposition algorithm for 2D and 3D geometric constraint solving. *Computer-Aided Design*, 38(1):1–13, 2006.

[10] J. E. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*. Graduate Studies in Mathematics. American Mathematical Society, 1993.

[11] C. M. Hoffmann and R. Joan-Arinyo. Symbolic constraints in constructive geometric constraint solving. *Journal of Symbolic Computation*, 23(2-3):287–299, 1997.

[12] C. M. Hoffmann and R. Joan-Arinyo. A brief on constraint solving. *Computer-Aided Design and Applications*, 2(5):655–663, 2005.

[13] C. M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *CP 1997: Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming*, pages 463–477, Hagenberg Castle, Austria, 1997.

[14] C. M. Hoffmann, M. Sitharam, and B. Yuan. Making constraint solvers more usable: overconstraint problems. *Computer-Aided Design*, 36(4):377–399, 2004.

[15] C. Jermann, B. Neveu, and G. Trombettoni. Algorithms for identifying rigid subsystems in geometric constraint systems. In *IJCAI '03: Proceedings of the 18th International Joint Conference on Aritificial Intelligence*, pages 233–238, Acapulco, Mexico, 2003. Morgan Kaufmann.

[16] C. Jermann, G. Trombettoni, B. Neveu, and P. Mathis. Decomposition of geometric constraint systems: a survey. *International Journal of Computational Geometry and Applications*, 16(5,6):379–414, 2006.

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

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

[19] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana-Pasto. Revisiting decomposition analysis of geometric constraint graphs. *Computer-Aided Design*, 36(2):123–140, 2004.

[20] P. Jörg, M. Sitharam, Y. Zhou, and J. Fan. Elimination in generically rigid 3D geometric constraint systems. In *Algebraic Geometry and Geometric Modeling*, Mathematics and Visualization, pages 205–216, Nice, France, 2004. Springer-Verlag.

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

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

[23] A. Mantler and J. Snoeyink. Banana spiders: a study of connectivity in 3D combinatorial rigidity. In *CCCG '04: Proceedings of the 16th Canadian Conference on Computational Geometry*, pages 44–47, Montréal, Québec, Canada, 2004.

[24] P. Mathis and S. E. B. Thierry. A formalization of geometric constraint systems and their decomposition. *Formal Aspects of Computing*, 22(2):129–151, 2010.

[25] D. Michelucci and S. Foufou. Interrogating witnesses for geometric constraint solving. In *SPM '09: Proceedings of the SIAM/ACM joint conference on Geometric and Physical Modeling*, pages 343–348, San Francisco, California, USA, 2009. ACM.

[26] D. Michelucci, S. Foufou, L. Lamarque, and D. Ménegaux. Another paradigm for geometric constraints solving. In *CCCG '06: Proceedings of the 18th Annual Canadian Conference on Computational Geometry*, pages 169–172, Queen's University, Ontario, Canada, 2006.

[27] A. Noort, M. Dohmen, and W. F. Bronsvoort. Solving over- and under-constrained geometric models. In B. Brüderlin and D. Roller, editors, *Geometric Constraint Solving and Applications*, chapter 2, pages 107–127. Springer, 1998.

[28] J.-J. Oung, M. Sitharam, B. Moro, and A. Arbree. FRONTIER: fully enabling geometric constraints for feature-based modeling and assembly. In *SMA '01: Proceedings of the 6th ACM symposium on Solid Modeling and Applications*, pages 307–308, Ann Arbor, Michigan, USA, 2001. ACM.

[29] J. C. Owen. Algebraic solution for geometry from dimensional constraints. In *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 397–407, Austin, Texas, United States, 1991. ACM.

[30] P. Schreck and P. Mathis. Geometrical constraint system decomposition: a multi-group approach. *International Journal of Computational Geometry and Applications*, 16(5,6):431–442, 2006.

[31] M. Sitharam. Well-formed systems of point incidences for resolving collections of rigid bodies. *International Journal of Computational Geometry and Applications*, 16(5,6):591–615, 2006.

[32] G. Trombettoni and M. Wilczkowiak. GPDOF: a fast algorithm to decompose under-constrained geometric constraints: Application to 3D modeling. *International Journal of Computational Geometry and Applications*, 16(5-6):479–511, 2006.

[33] H. A. van der Meiden and W. F. Bronsvoort. A non-rigid cluster rewriting approach to solve systems of 3D geometric constraints. *Computer-Aided Design*, 42(1):36–49, 2010.

[34] K. Weihrauch. *Computable analysis: an introduction*. Springer-Verlag New York, Inc., 2000.

[35] L. Yang. Solving geometric constraints with distance-based global coordinate system. In *International Workshop on Geometric Constraint Solving*, Beijing, 2003.

[36] C. Yap. *Fundamental problems in algorithmic algebra*. Oxford University Press, 2000.

[37] G.-F. Zhang and X.-S. Gao. Well-constrained completion and decomposition for under-constrained geometric constraint problems. *International Journal of Computational Geometry and Applications*, 16(5,6):18–35, 2006.