A Draft about Geometric Constraints Solving

L Lamarque and D Michelucci

February 8, 2006

Abstract

Geometric Constraints Solving use graph-based methods to decompose systems of geometric constraints. These methods have intrinsic and unavoidable limitations which are overcome by the witness method presented here.

1 Introduction

Geometric Constraints Solving [Owe91, Owe96, BR98, Hof06] today pervades all geometric modellers and geometric applications, in CAD-CAM, chemistry, robotics, virtual reality. Geometric constraints enable the designer to specify shapes with geometric constraints. Typical geometric constraints are specifications of distances, angles, incidences, tangencies, parallelisms, orthogonalities between geometric elements such as points, lines, planes, conics or quadrics, or higher degree algebraic curves and surfaces; a lot of problems in robotics (*e.g.,* generalized Stewart platform), in molecular chemistry (finding the configurations of a molecule from inter-atomic distances: it is the molecule problem), in geometric modelling (*e.g.,* blending surfaces), etc can be formulated with geometric constraints.

Systems of geometric constraints met in industry are bigger and bigger. For instance, geometric constraints begin to be used to define the control points of parameterized patches in 3D: there are 48 unknown coordinates per bicubic patch, and the simplest shape can require a dozen of such patches.

Thus decomposing such huge systems of geometric constraints into smaller subsystems is essential for solving. Today graph base methods [Owe91, Owe96] are used to decompose systems of geometric constraints, to planify the resolution of subsystems and to merge their solutions. A lot of graph based methods have been proposed; all rely more or less on a combinatorial count of degrees of freedom (DoF); they use graph flow computations, maximum matchings, k-connectedness properties.

These graph-based methods work very well for correct systems of constraints, and they indeed make possible to solve systems which are untractable otherwise. These methods are even able to detect the simplest mistakes in systems of constraints, namely structural dependences: a typical structural dependence

occurs each time a subset of unknowns is constrained by too much constraints. However, there are more subtle dependences, due to geometric theorems, which can not be detected with pure graph-based methods. Missed dependences make the solver fail, and the solver gives no usable explanation to the designer. This is a real problem: the probability of such dependences increases with the size of the systems to be solved, and with the availability and the increasing use of geometric solvers, actually with the success of GCS.

Of course, researchers in GCS are aware of this limitation of graph-based methods, and they agree that graph-based methods must be improved to better detect dependences between geometric constraints.

Unfortunately, detecting dependences with pure graph-based methods is an untractable problem (under the usual assumption that $P \neq NP$). It is due to the fact that every system of algebraic equations is translatable into a system of point-line incidences in the projective plane with a size of the same order of magnitude. Because of this universal property of systems of point-line incidences, detecting dependences between such constraints is as hard as detecting dependences between algebraic equations; no polynomial time method is known, and no one exists under the usual assumption that $P \neq NP$.

Such seemingly trivial incidence constraints between flats (points, lines, planes) are essential in real-world problems (the molecule problem being an exception). Thus there is no reasonable hope to make pure graph-based methods robust against subtle dependences since constraints include incidence constraints. However, not all is lost. It is possible to check the independence between the geometric constraints, to decompose them, or to check that a decomposition proposed by any other method is correct, assuming a witness configuration is available. For CAD-CAM problems, it is typically the case: systems of geometric constraints give a system of equations $F(U, X) = 0$ to solve, where U is a vector of parameters: specified distances or angle cosines, or non geometric values (weights, forces, costs); the value of U is known just before resolution. A witness is a couple (V, X_V) such that $F(V, X_V) = 0$, where $V \neq U$; in other words, a witness is a solution of a variant of the system to be solved. The witness and the target (the unknown, searched configuration) have the same properties, the same jacobian structure.

In the interactive setting, the sketch interactively provided by the user is often such a witness; points which must be aligned or coplanar in the solution are already so in the sketch, and only the angles and distances in the sketch need to be corrected by the solver. When a witness is not available (for instance in some batch use of geometric constraint solving), a witness is most of the time easily computable: consider the parameter U as unknowns and solve the very under-constrained $F(Y, X) = 0$. While witnesses can be arbitrarily difficult to find for some systems, due to the universality theorem, none of these systems seems relevant for CAD-CAM.

Foufou et al introduced the witness method [FMJ05]; they imposed useless limitations on the way to express constraints. This paper simplifies the method and broadens its scope. It provides detailed examples.

This paper is structured as follows. Section 2 present basic notions: free in-

finitesimal motions, displacements and flexions, degrees of displacements, how they are computed with rank considerations. Section 3 explains how the witness method answers to questions: are these constraints coordinates free? are they dependent or not? what is the minimal dependent set of equations? is a part rigid, *i.e.,* fixed by the system up to displacement? does the witness (and the target) have an unexpected geometric property, for instance a collinearity between 3 points? These capabilities of the witness method, combined with decomposition strategies of graph based methods, make possible more reliable and robust decomposition methods. Nevertheless, section 4 presents a first decomposition method which relies only on the witness. Section 5 argues about the difficulty of finding a witness: on one hand, this problem can be arbitrarily difficult; on the other hand, for CAD-CAM problems we always find easily a rational witness with rational coordinates. Section 5 also asks it is possible to "understand" equations, *i.e.,* to decompose and solve them without the knowledge of the underlying geometric constraints. Section 6 concludes. For completeness, appendix 7 presents the universality theorem, some typical geometric theorems which often confuse graph-based methods, and the molecule problem studied by the rigidity theory.

2 Free Infinitesimal motions

2.1 Definitions and notations

Assume a witness configuration (V, X_V) is known, *i.e.,* $F(V, X_V) = 0$. The main idea is to compute the vector $\dot{X}$ of the free infinitesimal motions $\epsilon \times \dot{X}$ of the witness, such that the perturbed witness $X_V + \epsilon\dot{X}$ still fulfils the specified constraints: $F(V, X_V + \epsilon\dot{X}) = 0$. With Taylor expansion, $F(V, X_V + \epsilon\dot{X}) = F(V, X_V) + \epsilon F'(V, X_V)\dot{X}^t + O(\epsilon^2)$. Thus $F'(V, X_V)\dot{X}^t$ must vanish: the free motions are given by the kernel of the jacobian matrice $F'(V, X_V)$ at the witness.

In all this paper, ϵ is just a symbol used for explanations; it is not represented in the computer, neither by a numerical value, nor by a symbol or any other data structure.

Free infinitesimal motions are usually classified in two classes: first infinitesimal displacements, namely translations, rotations and their compositions; and second infinitesimal flexions (sometimes called deformations), which deform the configuration. Clearly, if the witness admits such an infinitesimal flexion, the witness is flexible, *i.e.,* the system does not determine completely the geometric configuration.

A base of the infinitesimal displacements is computable a priori: it does not depend on the constraints, but only on the variables. Such a base is provided below; the infinitesimal scaling (which is not a displacement) is also provided.

There are several kinds of unknowns. We use the following conventions. In 2D, a point has coordinates (x, y); a line with equation $ax + by + c = 0$ is represented by a vector (a, b, c); a vector is represented by its coordinates (u, v); this distinction between points and vectors is due to the fact that a translation

(including an infinitesimal translation) modifies the (x, y) of points, but it does not modify the (u, v) of vectors, which are differences between points; similarly translations do not modify the a, b coefficients of lines, but they modify the c coefficient. For displacements, the variables u, v and a, b behave the same, but not for scaling. Other geometric unknowns (barycentric coordinates, scalar products, distances, squared distances, angle cosines, areas, volumes) are unchanged by infinitesimal displacements, so the corresponding entry in all vectors of the base are 0. The same holds for all non geometric unknowns (weights, costs, densities, temperatures...).

2.2 Basis of infinitesimal displacements

In 2D, a basis for the infinitesimal displacements is t_x, t_y, r_{xy}, where t_x is a translation in the x direction, t_y a translation in the y direction, and r_{xy} a rotation around the origin. Corresponding coordinates $\dot{x}$, $\dot{y}$, $\dot{b}$, $\dot{c}$, $\dot{u}$, $\dot{v}$ are given in this table:

	$\dot{x}$	$\dot{y}$	$\dot{a}$	$\dot{b}$	$\dot{c}$	$\dot{u}$	$\dot{v}$
t_x	1	0	0	0	$-a$	0	0
t_x	0	1	0	0	$-b$	0	0
r_{xy}	$-y$	x	$-b$	a	0	$-v$	u
s	x	y	$-a$	$-b$	0	u	v

The last line s describes the infinitesimal scaling (it is not a displacement).

Similarly, in 3D, a basis for the infinitesimal displacements is $t_x, t_y, t_z, r_{xy}, r_{yz}, r_{yz}$, where t_z is a translation along z, r_{yz}, r_{xz}, r_{xy} are rotations around the x, the y, the z axis. Corresponding coordinates are given in this table:

	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{a}$	$\dot{b}$	$\dot{c}$	$\dot{d}$	$\dot{u}$	$\dot{v}$	$\dot{w}$
t_x	1	0	0	0	0	0	$-a$	1	0	0
t_y	0	1	0	0	0	0	$-b$	0	1	0
t_z	0	0	1	0	0	0	$-c$	0	0	1
r_{xy}	$-y$	x	0	$-b$	a	0	0	$-v$	u	0
r_{xz}	$-z$	0	x	$-c$	0	a	0	$-w$	0	u
r_{yz}	0	$-z$	y	0	$-c$	b	0	0	$-w$	v
s	x	y	z	$-a$	$-b$	$-c$	0	u	v	w

A simple example in 2D is this typical system of 6 equations, with generic parameters δ (a distance) and λ (a cosine):

$$\begin{aligned}
e_1 &: ax + by + c = 0 \\
e_2 &: a'x + b'y + c' = 0 \\
e_3 &: (x - x')^2 + (y - y')^2 - \delta^2 = 0 \\
e_4 &: a^2 + b^2 - 1 = 0 \\
e_5 &: a'^2 + b'^2 - 1 = 0 \\
e_6 &: aa' + bb' - \lambda = 0
\end{aligned}$$

Fig.1 shows the jacobian and a base for infinitesimal motions: three displacements, and one flexion: the point (x', y') can rotate around the point (x, y)).

	x	y	x'	y'	a	b	c	a'	b'	c'
e'_1	a	b	0	0	x	y	1	0	0	0
e'_2	a'	b'	0	0	0	0	0	x	y	1
e'_3	$2(x-x')$	$2(y-y')$	$2(x'-x)$	$2(y'-y)$	0	0	0	0	0	0
e'_4	0	0	0	0	$2a$	$2b$	0	0	0	0
e'_5	0	0	0	0	0	0	0	$2a'$	$2b'$	0
e'_6	0	0	0	0	a'	b'	0	a	b	0
	$\dot{x}$	$\dot{y}$	x'	y'	$\dot{a}$	$\dot{b}$	$\dot{c}$	a'	b'	c'
t_x	1	0	1	0	0	0	$-a$	0	0	$-a'$
t_y	0	1	0	1	0	0	$-b$	0	0	$-b'$
r_{xy}	$-y$	x	$-y'$	x'	$-b$	a	0	$-b'$	a'	0
flexion	0	0	$y-y'$	$x'-x$	0	0	0	0	0	0

Figure 1: The jacobian and a base of infinitesimal motions: three displacements and a flexion.

The reader can check that the vectors for infinitesimal motions are orthogonal to the gradient vectors (the derivatives): $e'_1, \ldots e'_6$.

We prove only the basis for 2D infinitesimal displacements, the proof in 3D is similar. Let $P = (x, y, 1)$ be a point in homogeneous coordinates. P lies on a line $L = (a, b, c)$. If a displacement represented by a matrix M maps the point P to the point $P' = (x', y', 1) = PM$, and the line L to the line $L' = (a', b', c')$, then $L'^t = M^{-1}L^t$. Proof: $PL^t = 0$ and $P' = PM \Rightarrow PMM^{-1}L^t = 0 \Rightarrow P'M^{-1}L^t = 0$. Identify L'^t and $M^{-1}L^t$, then conclude.

For the infinitesimal translation $\dot{t_x}$ along x:

$$(x', y', 1) = (x + \epsilon, y, 1) = (x, y, 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \epsilon & 0 & 1 \end{pmatrix}$$

thus

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\epsilon & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ b \\ -\epsilon a + c \end{pmatrix}$$

thus $\dot{x} = x' - x = \epsilon x$, $\dot{y} = y' - y = 0$, $\dot{a} = a' - a = 0$, $\dot{b} = b' - b = 0$, $\dot{c} = c' - c = -\epsilon a$. Dividing by ϵ gives $\dot{t_x}$. Similarly, for the infinitesimal translation t_y along y.

For r_{xy}, the rotation around the origin with an infinitesimal angle:

$$(x', y', 1) = (x - \epsilon y, \epsilon x + y, 1) = (x, y, 1) \begin{pmatrix} 1 & \epsilon & 0 \\ -\epsilon & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and thus

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} 1 & -\epsilon & 0 \\ \epsilon & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a - \epsilon b \\ b + \epsilon a \\ c \end{pmatrix}$$

The difference between the identity matrice and the product of the two matrices is in $O(\epsilon^2)$, thus negligible in front of ϵ. Thus $\dot{x} = x' - x = -\epsilon y$,

$\dot{y} = y' - y = \epsilon x$, $\dot{a} = a' - a = -\epsilon b$, $\dot{b} = b' - b = \epsilon a$, $\dot{c} = c' - c = 0$, and dividing by ϵ indeed gives r_{xy}. Vectors (u, v) are difference between two points, and thus $(\dot{u}, \dot{v})$ strightforwardly follows for all infinitesimal displacements. Another way to compute a base of infinitesimal displacements is explained further.

2.3 Infinitesimal scaling

Computing the infinitesimal vector for scaling permits to detect if the proportions of a part are determined by the system, in other words if the shape is determined up to scaling. Contrarily to displacements, scaling modifies the lengths, areas, volumes and scalar products. Some systems well-constrained modulo displacements (*i.e.,* rigid) are still big and no more reducible into smaller rigid subsystems, but they are reducible into subsystems well-constrained modulo scaling (also called similitude). It is why infinitesimal scaling is relevant. Moreover, it highlights the difference between vectors u, v, w and normals a, b, c. The infinitesimal scaling maps $(x, y, 1)$ to

$$(x', y', 1) = (x, y, 1) \begin{pmatrix} 1+\epsilon & 0 & 0 \\ 0 & 1+\epsilon & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} 1-\epsilon & 0 & 0 \\ 0 & 1-\epsilon & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a - \epsilon a \\ b - \epsilon b \\ c \end{pmatrix}$$

The reader can check that the difference between the identity matrice and the product of the two matrices is in $O(\epsilon^2)$. Thus $\dot{x} = \epsilon x$, $\dot{y} = \epsilon y$, $\dot{a} = -\epsilon a$, $\dot{b} = -\epsilon b$, $\dot{c} = 0$, and divide by ϵ to simplify. If this vector is orthogonal to the gradient vectors of all equations, then the system is invariant through similitude (*i.e.,* scaling).

The scaling modifies the scalar products: If p is a scalar product of two vectors (u, v), then $\dot{p} = 2p$. If p is a scalar product of two normals (a, b), then $\dot{p} = -2p$. If p is a scalar product of a normal (a, b) and a vector (u, v), then $\dot{p} = 0$. The modifications for lengths, areas, follows. The extension to 3D is straightforward.

2.4 DoD: Degrees of Displacements

C. Jermann et al [JNT03] define degrees of rigidity to try to make graph-based methods more robust against dependences between constraints. We prefer the name: degrees of displacements. The degree of displacements of a rigid configuration (a set of points, lines, planes) is the number of equations needed to fix it in a cartesian coordinate system. The DoD is difficult to compute with pure graph based method. Jermann et al mainly suggest formulas for big enough configurations and a tabulation for a finite set of small configurations; moreover the configurations need to be "generic enough": incidences due to geometric

theorems are forbidden. This restriction is needed to avoid that the universality theorem applies.

The witness method computes straightforwardly the DoD, interrogating the witness. It can even precise which infinitesimal displacements are dependent, when they are. Let Y be the subset of the variables X which describes the configuration, and D be a base of the infinitesimal displacements at the witness. Then the DoD of Y is the rank of $D[Y]$. For instance, for a line (a, b, c) in 2D, $D[Y] = D[a, b, c]$ is:

	$\dot{a}$	$\dot{b}$	$\dot{c}$
t_x	0	0	$-a$
t_y	0	0	$-b$
r_{xy}	$-b$	a	0

and $D[Y]$ has rank 2: we can even see that it is the two translations t_x and t_y which are dependent. It is correct: a translation along the line leaves it unchanged. For the DoD of a segment $Y = (x, y, z, x'y', z')$ in 3D we just consider $D[Y]$ in the witness:

	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{x'}$	$\dot{y'}$	$\dot{z'}$
t_x	1	0	0	1	0	0
t_y	0	1	0	0	1	0
t_z	0	0	1	0	0	1
r_{xy}	$-y$	x	0	$-y'$	x'	0
r_{xz}	$-z$	0	x	$-z'$	0	x'
r_{yz}	0	$-z$	y	0	$-z'$	y'

which has rank 5; the 3 translations are independent; the three rotations are dependent, they have rank 2; it is correct: the rotation around the line supporting the segment leaves it unchanged. For the DoD of two secant planes $Y = (a, b, c, d, a', b', c', d')$ in 3D, we just consider $D[Y]$ at the witness:

	$\dot{a}$	$\dot{b}$	$\dot{c}$	$\dot{d}$	$\dot{a}$	$\dot{b}$	$\dot{c}$	$\dot{d}$
t_x	0	0	0	$-a$	0	0	0	$-a'$
t_y	0	0	0	$-b$	0	0	0	$-b'$
t_y	0	0	0	$-c$	0	0	0	$-c'$
r_{xy}	$-b$	a	0	0	$-b'$	a'	0	0
r_{xz}	$-c$	0	a	0	$-c'$	0	a'	0
r_{yz}	0	$-c$	b	0	0	$-c'$	b'	0

It has rank 5; more precisely, the three translations have rank 2, the three rotations are independent. The same way, we can compute the DoD of two parallel planes. Actually the interrogation of a witness gives us the DoD of any configuration. Absolutely no genericity assumption is required: the witness gives the right answer even when the configuration contains incidences which result from geometric theorems, for instance if three points are collinear due to Pappus theorem (or Desargues, or Pascal, or whatever). This power is due to the fact that the witness and the target share the same combinatorial properties. The computation of the rank of a set of (numerical only) vectors solves all problems.

When a part has DoD 3 in 2D and 6 in 3D, we say it has full DoD. This notion is used further for anchors.

2.5 Rank computations

To compute ranks, our program uses a variant of Gauss triangulation method: the latter uses only operations: $+, -, \times, \div$ and the nullity test; it does not need a non linear operation, in contrast to the Gram-Schmit orthogonalization procedure which needs the square root; thus Gauss method can be used in finite fields, for instance in $\mathbb{Z}/p\mathbb{Z}$ where p is a prime integer (close to 10^9), with exact arithmetic; it avoids inaccuracy problems which can pose difficulty when testing nullity, which is needed to compute the rank of a set of vectors. Of course, this choice needs that the witness has rational coordinates (exact algebraic arithmetic exist, but they are unconvenient and costly), and it is only a probabilistic test: there is a small probability ($1/p$ under reasonable assumptions) for a non zero number to be zero modulo p; this nullity test is crucial, it is used to detect if a vector lies in the vectorial space generated by another set of vectors. To gain more confidence in the nullity test, we redo the computation modulo several other primes: the probability for a non zero rational number to be equal to zero modulo all primes $p_1, p_2 \dots p_k$ is (under realistic assuptions) one over the product $p_1 p_2 \dots p_k$ of the primes.

Another choice is to compute with floating point numbers; the witness has floating point coordinates and need not to be rational, the square root operation and the Gram-Schmit procedure can be used, etc; it is more confortable for the programmer; but floating point arithmetic is not exact, and, though SVD computations can limit inaccuracy errors, computing the rank of a set of vectors is still problematic in case of dependence between vectors: some ϵ heuristic must be used. The ϵ heuristic decides that a number is equal to zero when it its absolute value is smaller than a prescribed threshold classically called ϵ (the latter ϵ has nothing to see with the previous ϵ used in the expansion $X_V + \epsilon \dot{X} ...$).

3 Witness interrogations

3.1 Are constraints coordinates-independent?

Correct geometric constraints are generally assumed to be independent of the cartesian frame; but if users are enabled to specify constraints, they can make mistakes. A constraint is not independent of the cartesian frame if it is not orthogonal to at least one of the vectors in the base of infinitesimal displacements. For instance, in 2D, the constraint: $x_M = 0$ is orthogonal to the vectors of the y translation and of the rotation around the origin, but not to the vector of the translation in x. Arbitrarily complicated equations can be tested this way. These tests are only numerical: the witness is a numerical vector, as the base of infinitesimal displacements in the witness. From now on, all equations are independent of the cartesian frame.

	x_O	y_O	x_A	y_A	x_B	y_B	x_C	y_C
e'_1	2	0	-1	0	-1	0	0	0
e'_2	0	2	0	-1	0	-1	0	0
e'_3	$2x_A - 2x_C$	$2y_A - 2y - C$	$2x_O - 2x_A$	$2y_O - 2y_A$	0	0	$2x_C - 2x_O$	$2y_C - 2y_O$
e'_4	0	0	$x_B - x_A$	$y_B - y_C$	$x_A - x_C$	$y_A - y_C$	$2x_C - x_A - x_B$	$2y_C - y_A - y_B$
e'_5	$2x_O - 2x_A$	$2y_O - 2y_A$	$2x_A - 2x_O$	$2y_A - 2y_O$	0	0	0	0

	$\dot{x_O}$	$\dot{y_O}$	$\dot{x_A}$	$\dot{y_A}$	$\dot{x_B}$	$\dot{y_B}$	$\dot{x_C}$	$\dot{y_C}$
t_x	1	0	1	0	1	0	1	0
t_y	0	1	0	1	0	1	0	1
r_{xy}	$-y_O$	x_O	$-y_A$	x_A	$-y_B$	x_B	$-y_C$	x_C
flexion	0	0	0	0	0	0	$y_O - y_C$	$x_C - x_O$

Figure 2: A dependent system. The jacobian, and a base of 4 free infinitesimal motions. The fourth is a flexion: point C can rotate around point O.

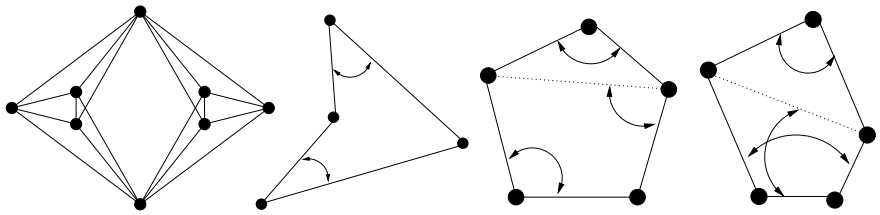

Figure 3: In 3D, the double banana, and three Ortuzar's configurations.

3.2 Are constraints dependent or independent?

Are the constraints dependent or independent? Graph based methods can detect only structural dependences, as in the system: $f(x, y, z) = g(z) = h(z) = 0$ which over-constrains the unknown z. Interrogation of the witness makes possible to detect more subtle dependences –actually all dependences– as follows.

The constraints are dependent if the gradient vectors of the equations at the witnes, *i.e.,* the jacobian matrice at the witness, are dependent. It suffices to compute a base of this jacobian, either with Gauss triangulation method, or with the Gram-Schmit orthogonalization procedure, or a LU factorization, or any other method in linear algebra (SVD for instance). All that is standard numerical linear algebra.

Typical and simple examples of 3D configurations where the witness method detects the dependence and the graph-based methods do not, are given in Fig. 3. The leftmost configuration is classical, and is known as the double banana. The dependence in double banana was already detected by a classical numerical probabilistic method [JG93], which the witness method encompasses.

It is possible to taylor pure graph-based methods to make them detect some of these dependences (*e.g.,* [Owe96, JNT03]). However this approach is terribly difficult, it can result in hairy methods and software, not easily reproducible. And, finally, the universal theorem makes untractable the problem of detecting all dependences with pure graph-based methods. However, the knowledge and

skills acquired with graph-based method is not useless: decomposition strategies in graph-based methods can be reused with the witness method; conversely, it is also possible to start from an existing graph-based method, and to check the correctness of the decomposition with the witness method.

3.3 Example of a dependent system

Here is a simple example of a dependent system. In 2D, point O is the middle of points A and B. Distance OC equals distance OA. AC and BC are orthogonal; this last constraint results from the previous ones, due to this geometric theorem: if C lies on the circle with diameter AB, then AC and BC are orthogonal. Distance OA is specified with a parameter u. The system of equations is

$$
\begin{aligned}
&e_1 : 2x_O - x_A - x_B = 0\\
&e_2 : 2y_O - y_A - y_B = 0\\
&e_3 : (x_C - x_O)^2 + (y_C - y_O)^2 - (x_A - x_O)^2 - (y_A - y_O)^2 = 0\\
&e_4 : (x_C - x_A)(x_C - x_B) + (y_C - y_A)(y_C - y_B) = 0\\
&e_5 : (x_A - x_O)^2 + (y_A - y_O)^2 - u^2 = 0
\end{aligned}
$$

Fig. 2 displays the jacobian and a base of the free infinitesimal motions: three displacements and a flexion, point C can rotate around point O. The rank of $e'_1, \ldots e'_5$ is computed at the witness, it is 4, thus equations are dependent.

3.4 Minimal dependent set of constraints

If the constraints are dependent, interrogation of the witness makes possible to find the smallest dependent set of constraints: this information is relevant for the user, who can fix more easily the mistake in the system of constraints (remember that constraints can be numerous). This problem reduces to finding the minimal dependent set in a dependent set of vectors (they are the gradient vectors of the equations at the witness). We assume that the rank of the dependent set is its cardinal minus one: typically, the last vector we try to add in the base reduces to the null vector. In such a case, the minimal dependent set is unique; to find it, just try to remove each vector in the dependent set; if the set minus this vector is still dependent, then remove this vector. This greedy method can be proved with matroid theory.

3.5 Rigidity test

Is the system flexible? Just compute a base of the kernel of the jacobian, at the witness: it is a base of the free infinitesimal motions of the witness. If it contains vectors outside the base of the infinitesimal displacements, then the witness (and the target) is flexible. For instance, in the classical configuration of the double banana, the two bananas can rotate around the axis through their two common vertices; the corresponding infinitesimal flexion is detected by the method.

If the system is flexible, the witness method can provide a base of the infinitesimal deformations, and the set of maximal rigid subparts (well-constrained modulo displacements), see below.

Is a part rigid? A flexible system can contain rigid parts. A part is described by a subset Y of the unknowns. On tables, each variable corresponds to a column, and a part Y is thus a subset of columns. The part Y is rigid iff the vectorial space $M[Y]$ (the free infinitesimal motions in the columns Y) is equal to the vectorial space $D[Y]$ (the free infinitesimal displacements in the columns Y). Vectors generating $M[Y]$ are obtained by taking only the columns Y in the vectors of the base of M. Similarly for $D[Y]$.

For instance, in the example of Fig.1, $Y = \{x, y, a, b, c, a', b', c'\}$ is rigid, though $Y \cup \{x', y'\}$ is flexible. It does not depend on the base chosen for M and D.

Are A and B relatively fixed? A flexible system can fix some pairs of geometric elements (two points, two lines, one point and one line, etc) relatively to each other. Actually, the previous section already provides a decision procedure. A and B are relatively fixed by the (possibly flexible) system if the part $Y = A \cup B$ is rigid.

3.6 Witnesses and probabilistic proofs

The witness makes possible to detect geometric coincidences: collinearity of 3 points, coplanarity of 4 points, parallelism, etc, which are due to geometric theorems (Pappus, Desargues, Pascal, etc) or to accidents. It permits probabilistic proofs [Mar71, Sch80], in the following sense: when a coincidence is due to a theorem, then the theorem is satisfied in the witness. It happens, very unlikely, that the witness is not generic enough and has accidentally a property. Each time a coincidence is detected, the test is performed again (modulo another prime integer), to achieve a greater confidence. The probability of wrong positives is $1/p$ if computations are done modulo p (in practice $P \approx 10^9$). With k such independent tests, the probability is one over the product of the used primes, say 10^{9k}. This kind of proofs can be made fully deterministic, with exponential cost: no magy.

Suppose it is conjectured that $C(X) = 0$ is a consequence of the system $F(X) = 0$. First check that the conjecture indeed holds in the witness, *i.e.,* that $C(X_V) = 0$. If not, the conjecture is clearly wrong. If $C(X_V) = 0$, then check that $C(X_V + \epsilon\dot{X}) = 0$ it is still true for all vectors $\dot{X}$ in the base of the free motions of the system $F(X) = 0$ at the witness X_V. Using Taylor as usual, $C(X_V + \epsilon\dot{X}) = C(X_V) + \epsilon C'(X_V)\dot{X}^t + O(\epsilon^2)$, thus the gradient vector $C'(X_V)$ must be orthogonal to $\dot{X}$ (in other words, $C'(X_V)$ must lie in the vectorial space spanned by the jacobian $F'(X_V)$). This protocol avoids to prove that all parallelograms are rectangles (or squares) when the witness should only be a parallelogram, and it is accidentally a rectangle (or a square).

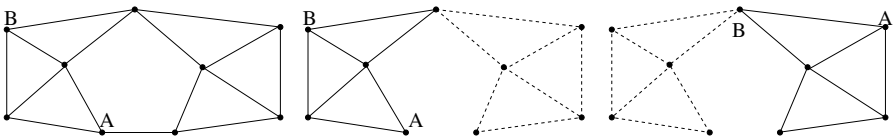

Figure 4: A 2D rigid system of constraints. Removing a constraint creates a flexible system with two MRP. In the 3 figures, the MRP(A, B) is in thick lines.

It is possible to consider terms in $\epsilon^2, \epsilon^3 \ldots$, *i.e.,* to compute series: $X_V + \epsilon\dot{X} + \epsilon^2\ddot{X} + \ldots$ (in this wake, computer algebra uses Puiseux expansions to study singularities). In our context, linear numerical algebra is still sufficient, but Hessians are more space consuming than Jacobians. We do not investigate this track. Maybe relevant informations can be extracted from $\ddot{X}$ (accelerations).

4 Witness-based decomposition

The previous decision procedures (is a part flexible or rigid?) are sufficient, when combined to any graph-based methods, to decompose systems. Notwithstanding, this section provides the first decomposition method which relies only on the witness, and which considers no graph at all. It considers only the array of the jacobian and the base of free infinitesimal motions. It tries to get rid of geometric considerations: it should make methods simpler and more general; too much often, methods based on the geometric intuition get stuck in a morass of geometric cases.

The witness-based decomposition method works as follows: if the configuration is flexible, it finds its maximal rigid parts (MRP). If the configuration is rigid, it removes each constraint in turn, to make it flexible. Some book keeping avoids to find several times the same MRP.

4.1 Finding an anchor

An anchor $Y \subset X$ is a part which is rigid and which has full DoD: the vectorial space of its free motions is equal to the vectorial space of the infinitesimal displacements (rank 3 in 2D, 6 in 3D); moreover the anchor has minimal cardinality, either in a geometric sense: the anchor is a set of geometric elements (points, lines, planes), or in an algebraic sense: the anchor is a set of variables. In both cases, these anchors are used as an argument for the procedure below, computing the maximal rigid part containing an anchor.

In 2D, a geometric anchor can be two points the distance of which is fixed by the system, but it can not be only one point. It can also be made of a line and a non incident point. It can also be made of three secant non concurrent lines, etc. In 3D, a geometric anchor can be 3 points, the three distances of which are fixed by the system. It can not be a segment: we saw that the DoD of a segment in 3D is 5, not 6. Clearly a configuration contains only a polynomial number of

geometric anchors. Every geometric anchor contains an algebraic anchor.

An algebraic anchor contains $r = 3$ variables in 2D, and $r = 6$ in 3D. Clearly there is a polynomial number $O(|X|^r)$ of potential algebraic anchors in the system with variables X ($|X|$ is the cardinal of X). For example, in 2D, a possible anchor is the set of variables x_1, y_1, x_2, iff the distance between the points (x_1, y_1) and (x_2, y_2) is fixed by the system (directly, by a constraint distance, or indirectly). Similarly, in 3D, a possible anchor is the set of variables $x_1, y_1, z_1, x_2, y_2, z_3$ if the three distances are fixed (directly or indirectly) by the system. The previous definition of algebraic anchors is geometrically counter intuitive, since it breaks geometric elements into variables. Anyway, the MRP method works for any kinds of anchors, and it will merge again broken geometric elements (points, lines or planes).

4.2 Finding the maximal rigid part MRP(Y)

Assume the system $F(X) = 0$ is flexible, and that the given part Y is rigid and has full DoD. Then Y is contained in a unique maximal rigid part, which has variables R. R is computed with the following greedy method: initialize R with Y, and for each variable $x \in X - Y$, if $Y \cup x$ is rigid, then insert x in R (the test: if $Y \cup x$ is rigid, can be replaced by the test: if $R \cup x$ is rigid).

Remark: the algorithm for the MRP(Y) extends to find the maximal similar part MSP(Y), *i.e.,* the maximal part determined up to scaling and containing a part Y determined up to scaling.

4.3 Finding all MRP

The set Ω of all maximal rigid parts is initialized to $\emptyset$. For all potential anchors A, if A is not already included in a MRP in Ω, then insert MRP(A) in Ω. This method works for both algebraic and geometrical anchors. The number of MRP is polynomial: there is only one MRP per anchor, and the number of anchors is polynomial. Thus this method is polynomial time. In passing, the number of MRP is much smaller than the number of anchors.

5 Miscellaneous remarks

5.1 Infinitesimal displacements

There is another way to compute a base of infinitesimal displacements, which requires less knowledge, and applies also for exotic systems of coordinates (for instance Grassman-PlÃ¼cker coordinates) not considered here. We only give the principle, skipping details. Add geometric constraints to make the configuration rigid; for example, for two points A and B which are distant by l in the witness, add the constraint: $(A - B).(A - B) - l^2 = 0$ (replacing l^2 by its numerical value in the witness). The fact that the augmented system is redundant does not matter. When the augmented system is rigid, compute a base of the kernel of its jacobian; it contains only displacements. If the base has not the good

rank (6 in 3D, 3 in 2D), the initial system is wrong; for instance, an initial equation depends on the coordinate system, or the system does not fix some non geometric unknowns.

5.2 Triangular inequalities

It can happen that the system has a witness and the solver finds no real solution. When the solver is complete (for instance, it is an interval based solver, which finds all existing real solutions), it means that the parameters of the system have bad values; for instance they violate a triangular inequality. Detecting which inequalities are violated by parameters is a problem of Real algebra and Real geometry; this kind of problems are theoretically solvable after Tarski, but untractable because of complexity.

5.3 Generating witnesses; which difficulty?

For the molecule problem, finding a witness is completely trivial: just generate random points, in 2D or in 3D, according to the problem. For general geometric constraints (including incidences), we can often use a dual method for generating a witness, for instance when the unknown configuration is a 3D polyhedra described by the length of its edges.

The octahedron problem, solved by Durand and Hoffmann, also known as the Stewart platform, and the icosahedron problem are just molecule problems: they have triangular faces, so generating random vertices is sufficient; the fact that the generated polyhedon is likely concave and even self intersecting does not matter, as far as the distance and coplanarity conditions are satisfied.

The hexahedron (6 quadrangular planar faces) or the dodecahedron (12 pentagonal planar faces) are examples of systems of geometric constraints, where the dual method works: generate random planes in 3D, one random plane per face, before computing the resulting vertices as intersection points of the supporting planes; it provides a witness. This method for generating a witness clearly relies on the fact that each vertex of the hexahedron and of the dodecahedron is degree 3. For vertices with greater degree, the method will not work, because there is a probability 0 for four (or more) random planes to meet in a common point.

Here, the good news is Steinitz's theorem [RG96]: all 3D polyhedra are realizable with rational numbers, and thus with integers (just multiply by the gcd of all denominators); it is worth to mention that, in contrast, 4D polyedra are not all realizable with integer coordinates [RG96]; thus we are lucky to live in a 3D space. Steinitz's theorem confirms the intuition that, when choosing a good set of base points and assigning them integer coordinates, finding the coordinates for the other points reduces to solving a linear system of equations. Admittedly, Steinitz's theorem and this intuition are only plausible arguments for feasibility, and not an algorithm; this is future work.

Sometimes, we need a bit more than a polytope for the witness and the target, something like several related polytopes, or a partially specified arrange-

ment (*i.e.,* a set of point-line or point-plane incidences). Due to the universality theorem, finding a realization (with integers or not) can be arbitrarily difficult, even in 2D. Up to now, we were always able to find a rational witness (the counter-example in [FMJ05] is not relevant fo r CAD-CAM), so we conjecture that difficult problems are not relevant for CAD-CAM, and that it is even easy to find a rational witness. It is fair to admit we did not consider problems of industrial size. The complexity of finding witnesses for CAD-CAM problems is an open issue.

5.4 Understanding equations

This paper assumes as usual that the geometric constraints and the system of equations are both available, *i.e.,* for each unknown in the system of equations, we know its semantic, for instance that it is the ordinate of such point. The two related questions:

- is it possible to decompose a system of equations, given the equations and a witness, without an a priori knowledge of the geometric constraints and of the meaning of the variables?

- is it possible to reconstruct geometric constraints from the system of equations only (a kind of reverse engineering) and a witness?

have not been posed so far. In other words, is it possible to "understand" equations? Assume the answer to both questions is positive. Then we can provide more powerfull solvers which accept as input systems of equations, rather than systems of constraints; systems of equations are much easier to standardize than systems of geometric constraints; actually MathML already does the job. Maybe also that decomposition methods would apply to more general systems of equations? To suggest that the underlying problem is relevant, and perhaps feasible, consider the problem of reconstructing geometric constraints and meanings of unknowns from a system of algebraic equations and a witness, in the 2D case, and when the system is rigid. We observe that, whatever the computed base for infinitesimal free displacements, $D[u_1 \ldots, v_1 \ldots, a_1 \ldots, b_1 \ldots]$ has rank 1; $D[x_1 \ldots, y_1 \ldots]$ has rank 2; for non geometric unknows $t_1 \ldots$, the rank of $D[t_1 \ldots]$ is 0. This makes possible to recover the meaning of each variable (up to symmetries, such as the symmetry between x and y).

6 Conclusion

Graph-based methods for decomposing systems of eqautions or constraints have intrinsic and unavoidable limitations; the witness method overcomes succesfully these limitations. This paper simplifies and generalizes the witness method. It makes arise two questions.

For CAD-CAM problems, is there always, or most of the time, a witness with rational coordinates? When a witness has rational coordinates, it is possible and cheap to perform exact computations in the rationals or modulo some finite

field, and this avoids the inaccuracy of the floating point arithmetic –and the excessive cost of algebraic (non rational) arithmetics.

Given only a system of equations and a witness, is it possible to "understand" the system of equations, *i.e.,* to extract its relevant properties and use them to decompose and solve it? We feel the witness method can still be simplified, and generalized to other kinds of invariance, at least to invariance modulo scaling, and modulo homography. Several things (for instance the duality between the maximal rigid part and the minimal dependent system) suggest that there is a deeper, simpler and more powerful theory for decomposition.

References

[BR98] Bruderlin B., Roller D. (Eds.): *Geometric Constraint Solving and Applications*. Springer-Verlag, 1998.

[CH88] Crippen G. M., Havel T. F.: *Distance Geometry and Molecular Conformation*. Research Studies Press, Taunton, U.K., ISBN 0-86380-073-4, 1988.

[FMJ05] Foufou S., Michelucci D., Jurzak J.-P.: Numerical decomposition of geometric constraints. In *ACM Symp. on Solid and Physical Modelling* (2005), pp. 143–151.

[Hen92] Hendrickson B.: Conditions for unique realizations. *SIAM J. Computing 21*, 1 (feb 1992), 65–84.

[Hof06] Hoffmann C. M.: Summary of basic 2d constraint solving. *International Journal of Product Lifecycle Management 1*, 2 (2006), 143 – 149.

[JG93] J. Graver B. Servatius H. S.: *Combinatorial Rigidity. Graduate Studies in Mathematics*. American Mathematical Society, 1993.

[JNT03] Jermann C., Neveu B., Trombettoni G.: Algorithms for identifying rigid subsystems in geometric constraint systems. In *IJCAI* (2003), pp. 233–238.

[Mar71] Martin W.: Determining the equivalence of algebraic expressions by hash coding. *J. ACM 18*, 4 (1971), 549–558.

[Owe91] Owen J.: Algebraic solution for geometry from dimensional constraints. In *Proc. of the Symp. on Solid Modeling Foundations and CAD/CAM Applications* (1991), pp. 397–407.

[Owe96] Owen J.: Constraint on simple geometry in two and three dimensions. *Int. J. Comput. Geometry Appl. 6*, 4 (1996), 421–434.

[RG96] Richter-Gebert J.: *Realization Spaces of Polytopes*. Lecture Notes in Mathematics 1643, Springer, 1996.

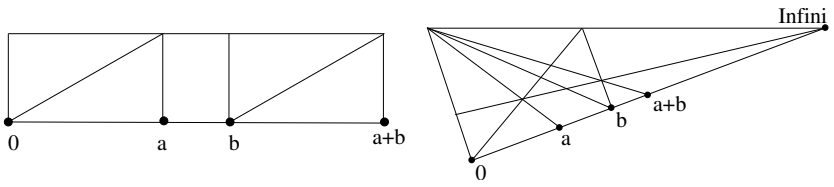

Figure 5: Affine and projective construction of $a + b$.

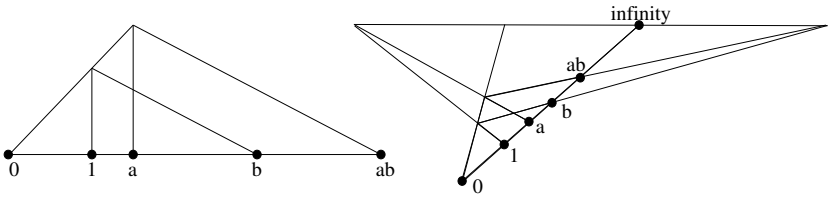

Figure 6: Affine and projective construction of $a \times b$.

[Sch80] SCHWARTZ J.: Fast probabilistic algorithms for verification of polynomial identities. *J. ACM 4*, 27 (1980), 701–717.

7 Appendix

7.1 The universality theorem

The algebraic constraint : $z = x+y$ can be translated into a set of geometric constraints in the Euclidean plane between points and lines : point-line incidences and constraints of parallelism between lines, with the geometric construction shown in fig. 5. Similarly for the algebraic constraint: $z = x \times y$, with the geometric construction shown in fig. 6. Actually, parallelism constraints between lines reduce to point-line incidence constraints too, introducing an auxiliary and arbitrary line, called the line at infinity by Desargues: the constraint: lines l and l' are parallel, is replaced by the constraint: lines l, l', and the line at infinity concur. Then every algebraic system can be translated into a set of point-line incidence constraints: coefficients, unknowns, monomials, polynomials are all represented by points on a given line; from two arbitrary, distinct, basic points 0 and 1 on this line, all points representing an integer n (for instance a coefficient in the algebraic system) can be built with $O(\log n)$ constraints, using iterated squaring. Iterated squaring is also used to generate the constraints involving monomials. This guarantees that the system of point-line incidences and the algebraic system have a bit size of the same magnitude.

7.2 Some geometric theorems

This appendix lists some geometric theorems which confuse graph-based methods (for the moment, no theorem is known which confuses the witness method).

Desargues theorem holds in 2d and in 3D. If two triangles $p_1p_2p_3$ and $q_1q_2q_3$ are perspective (*i.e.,* the three lines p_iq_i concur), then the three intersection points $r_{ij} = p_ip_j \cup q_iq_j$ are aligned.

Pappus theorem: in 2D, if points p_1, p_2, p_3 are aligned, as well as q_1, q_2, q_3, the three intersection points r_{12}, r_{23}, r_{13} where $r_{ij} = p_iq_j \cap p_jq_i$ are aligned as well.

Pappus dual theorem: two triangles which are perspective in two ways are perspective in three ways.

Pascal theorem, or Pascal mystical hexagram: in 2D, if points $p_1, p_2, p_3, q_1, q_2, q_3$ lie on a common conic, then the three intersection points $r_{ij} = p_iq_j \cap p_jq_i$ are aligned. Tersely: the opposite sides of an hexagon inscribed in a conic meet on three aligned points. Pappus is a consequence of Pascal: two lines are a degenerate conic.

Brianchon theorem is the dual of Pascal theorem. When a conic is inscribed in an hexagon, the three diagonals of the hexagon concur.

Hexamys theorem: an hexamys is an hexagon, such that the opposite sides meet on three aligned points. Then every permutation of an hexamys is an hexamys. It is a reformultation of Pascal theorem.

Chasles theorem: if points $p_1 \dots p_9$ are the 9 intersection points between two cubic curves without a common component, then every cubic curve passing through 8 of 9 points also passes through the 9th one.

The three quadrics theorem: if $p_1, \dots p_8$ are the eight intersection points between three quadrics (without pairwise common component), then every quadric through seven of these eight intersection points also goes through the 8th one.

Beltrami theorem: in 3D, assume that three black non pairwise coplanar lines b_i cut three white non pairwise coplanar lines w_j in 9 points. Then every line cutting the three black lines and every line meeting the three white lines are coplanar (*i.e.,* meet).

7.3 The molecule problem

In the molecule problem [JG93, Hen92, CH88], the sole geometric constraints are distances between points. When these distances are generic, the combinatorial characterization of rigidity (well-constrainedness modulo isometry, or displacement) is known for the 2D case: it is Laman's theorem. It states that a system of c point-point distances in 2D, between n points, is well-constrained modulo displacement iff $c = 2n - 3$ and there is no over-constrained subsystem, *i.e.,* no subsystem of n' points and c' distance constraints where $c' > 2n' - 3$. Though there is an exponential number of subsystems to test, several polynomial time methods were proposed [JG93, Hen92]. Graph-based decomposition methods used in GCS are clearly extensions of these methods.

The extension of Laman's theorem to 3D (replacing $2n - 3$ by $3n - 6$) gives a necessary condition for rigidity, but it is not sufficient. The double banana is probably the most famous counter-example. The combinatorial characterization of rigidity in 3D and beyond is today an active research area in combinatorics, especially matroid theory; though this characterization is still unknown, there

is a classical numerical and probabilistic method to test rigidity, in polynomial time (in cubic time): it is the witness method. The correctness of the witness method for the molecule problem is Gluck's theorem [JG93]. For the molecule problem, finding a witness is completely trivial: just generate random points. The genericity condition prevents collinearity, coplanarity, cocyclicity, etc. This condition is essential for rigidity theory; otherwise non generic distances enable incidence constraints and the universal theorem applies. Unfortunately, this condition is incompatible with CAD-CAM applications.