Abstract - Geometric constraint solving has applications in many different fields, such as Computer-Aided Design, molecular modelling, tolerance analysis, and geometric theorem proving. Geometric modelling by constraints enables users to describe shapes by relationships called constraints between geometric elements. The aim is to derive automatically these geometric elements and provide thus effort and time saving. Moreover, users can easily modify existing designs. Many resolution methods have been proposed for solving systems of geometric constraints. We classify these methods in three broad categories: algebraic, rule-oriented and graph-constructive solvers.

Keywords - Constraints solving, geometric constraints, algebraic, rule-oriented and graph-constructive solvers

I. INTRODUCTION

Geometric constraint solving has applications in many different fields, such as Computer-Aided Design, molecular modelling, tolerance analysis, and geometric theorem proving. Geometric modelling by constraints enables users to describe shapes by specifying a rough sketch and adding to it geometric constraints i.e. a set of required relations between geometric elements (see [6], [27], [28]). The constraint solver must derive automatically the correct shape needed. Typically, in 2D, geometric modelling by constraints specifies geometrical objects such as points, lines, circles, conics by a set of constraints: distances between points, points and lines, parallel lines, angles between lines, incidence relations between points and lines, points and circles, tangency relations between lines and circles or between circles, and so on.

Intuitively a dimensioned sketch is considered to be well-constrained if it has a finite number of solutions for non-degenerate configurations. Figure 1 shows a 2D well-constrained problem with three distance constraints between pairs of points and two angle constrains. Similarly a dimensioned sketch is considered to be under-constrained if it has an infinite number of solutions for non-degenerate configurations. Finally a dimensioned sketch is considered to be over-constrained if it has no solutions for non-degenerate configurations. Examples of 2D under-constrained and over-constrained sketches are given in figures 2.a and 2.b respectively.

Many resolution methods have been proposed for solving systems of geometric constraints. They can be classified in many ways. We classify these methods in three broad categories: algebraic, rule-constructive and graph-constructive solvers.

This paper is organized as follows. The algebraic approach for constraints solving is explained in section II. We present in section III, the rule-oriented methods to handle constraint problems. The graph-based methodology is given in section IV. Section V gives conclusions.

II. ALGEBRAIC APPROACH

A. Numerical methods

In a numerical constraint solver, the geometric constraints are first translated into a system of algebraic equations (linear or not). This system is then solved by applying a numerical method. For each constraint correspond an algebraic equation. As an example of this correspondence, the equation corresponding to “internal”
and "external" tangency between two circles (see figure 3), one centred at $A$ with radius $r_1$, the second centred at $B$ with radius $r_2$ is the following:

$$(x_B - x_A)^2 + (y_B - y_A)^2 - (r_1 \pm r_2)^2 = 0.$$  

Example:

A constrained scheme is shown in figure 4. It consists of three points and three distance constraints. The point $P_1$ lies at the origin and $P_3$ on the positive x-axis. The corresponding set of equations is given below:

$$\begin{align*}
eq_1 : x_1 &= 0. \\
eq_2 : y_1 &= 0. \\
eq_3 : y_3 &= 0. \\
eq_4 : (x_2 - x_1)^2 + (y_2 - y_1)^2 - d_1^2 &= 0. \\
eq_5 : (x_3 - x_2)^2 + (y_3 - y_2)^2 - d_3^2 &= 0. \\
eq_6 : x_3 - x_1 - d_2 &= 0.
\end{align*}$$

Sutherland [55], Hillyard and Braid [24], Borning [7] Borning and Duisberg [8] and Gosling [22] have used the relaxation method to solve the constrained problem. Relaxation is a quite slow method and sometimes convergence can't be achieved. Barford [5] used a projective method.

Many systems have used the Newton-Raphson's iteration to solve the set of geometric constraints. This "popular" numerical method was used, among many others, by Lin et al. [41], Light et al. [40], Lee et al. [37], Nelson [45], Prusinkiewicz et al. [49], Rocheleau et al. [50], Gossard et al. [23], Serrano [51], Perez et al [47], Anderl et al. [4]. This method needs an initial guess, typically given by the sketch of the desired geometric scheme.

However, there is a well-known problem. If Newton-Raphson’s method often works well, sometimes it does not converge or it converges to an unwanted solution. In this last case, the user changes his initial guess until Newton-Raphson’s method works if it does.

An alternative method to Newton-Raphson for geometric constraint solving is homotopy or continuation [3]. Lamure et al. [34] have tested several configurations using homotopy with success and where Newton-Raphson's method fails. Nonetheless, homotopy, is slower than Newton's method. Homotopy was also used for solving 3D constraints by Durand et al. [14]. Foufou et al. [16] recently used efficient numerical methods to solve systems of constraints.

B. Symbolic methods

As in the numerical solvers, the constraints are again formulated as a system of algebraic equations. However, instead of applying numerical techniques to determine a solution, general symbolic computations are undertaken to find the solution to the system of equations. Methods such as Gröbner basis or Wu-Ritt [32] techniques can be applied to find symbolic expressions for the solutions.

Many systems have used a symbolic resolution of the system of equations. We can non exhaustively mention works done by Ericson et al. [15], Kondo [31], Buchanan et al. [12], Chou et al. [13] and Gao et al [18].

III. RULE-CONSTRUCTIVE APPROACH

A. Problem formulation

Rule-based solvers rely on the predicates formulation. The constraints are expressed as facts and an inference engine is used to derive the solution by exhaustively applying rules.

Some examples of expressing constraints as facts are given below:

$$\begin{align*}
\text{coord}(P_1, [X, Y]) & \text{ coordinates of point P1 are X and Y.} \\
\text{dist}(P_1, P_2, d) & \text{ d is the distance between points P1 et P2.} \\
\text{angle}(P_1, P_2, P_2, b) & \text{ b is the angle P1P2P3.} \\
\text{slope}(P_1, P_2, a) & \text{ a is the slope of the line P1P2.} \\
\text{slope}(P_1, P_2) & = \text{slope}(P_3, P_4) \text{ line P1P2 is parallel to line P3P4.}
\end{align*}$$

A typical example of rule used by the inference engine is given hereafter:

$$\begin{align*}
\text{coord}(P_1, [X_1, Y_1]), \text{coord}(P_2, [X_2, Y_2]), \text{dist}(P_1, P_2, R_1), \\
\text{dist}(P_2, P_3, R_2) & \rightarrow \text{coord}(P_3, \text{intersection(circle}(c(P_1), R_1), \text{circle}(c(P_2), R_2))))
\end{align*}$$

However, there is a well-known problem. If Newton-Raphson’s method often works well, sometimes it does not converge or it converges to an unwanted solution. In this last case, the user changes his initial guess until Newton-Raphson’s method works if it does.

An alternative method to Newton-Raphson for geometric constraint solving is homotopy or continuation [3]. Lamure et al. [34] have tested several configurations using homotopy with success and where Newton-Raphson's method fails. Nonetheless, homotopy, is slower than Newton's method. Homotopy was also used for solving 3D constraints by Durand et al. [14]. Foufou et al. [16] recently used efficient numerical methods to solve systems of constraints.

B. Symbolic methods

As in the numerical solvers, the constraints are again formulated as a system of algebraic equations. However, instead of applying numerical techniques to determine a solution, general symbolic computations are undertaken to find the solution to the system of equations. Methods such as Gröbner basis or Wu-Ritt [32] techniques can be applied to find symbolic expressions for the solutions.

Many systems have used a symbolic resolution of the system of equations. We can non exhaustively mention works done by Ericson et al. [15], Kondo [31], Buchanan et al. [12], Chou et al. [13] and Gao et al [18].

III. RULE-CONSTRUCTIVE APPROACH

A. Problem formulation

Rule-based solvers rely on the predicates formulation. The constraints are expressed as facts and an inference engine is used to derive the solution by exhaustively applying rules.

Some examples of expressing constraints as facts are given below:

$$\begin{align*}
\text{coord}(P_1, [X, Y]) & \text{ coordinates of point P1 are X and Y.} \\
\text{dist}(P_1, P_2, d) & \text{ d is the distance between points P1 et P2.} \\
\text{slope}(P_1, P_2, a) & \text{ a is the slope of the line P1P2.} \\
\text{angle}(P_1, P_2, P_2, b) & \text{ b is the angle P1P2P3.} \\
\text{slope}(P_1, P_2) & = \text{slope}(P_3, P_4) \text{ line P1P2 is parallel to line P3P4.}
\end{align*}$$

A typical example of rule used by the inference engine is given hereafter:

$$\begin{align*}
\text{coord}(P_1, [X_1, Y_1]), \text{coord}(P_2, [X_2, Y_2]), \text{dist}(P_1, P_2, R_1), \\
\text{dist}(P_2, P_3, R_2) & \rightarrow \text{coord}(P_3, \text{intersection(circle}(c(P_1), R_1), \text{circle}(c(P_2), R_2))))
\end{align*}$$
It states that if points $P_1$ and $P_2$ are known and the distance values $R_1$ and $R_2$ between points $(P_1, P_3)$ and $(P_2, P_3)$ respectively are given then we can compute the coordinates of the point $P_3$ by intersecting two circles, one centered at $P_1$ with radius $d_{13}$ the other centered at $P_2$ with radius $d_{23}$.

B. Rule-Oriented solvers

We will review now some works done in the rule-oriented field. The majority of rule-constructive solvers have been implemented at the end of the eighties and early nineties.

Brüderlin ([10], [11]), has developed a system taking into account 2D and 3D constraints. This system was implemented using Prolog. The class of 2D configurations solved was not very large and the 3D case solves very simple schemes.

Sunde in [53] uses a rule-constructive method. He introduces the notion of CA (Constraint Angle) and CD (Constraint Distance) sets. The method tries to merge all the initial CA and CD sets in a single CD set. If the merging operation succeeds the figure solution is constructed.

Aldefeld in [2] uses a forward chaining inference mechanism, where the notion of direction of lines is imposed by introducing additional rules, and thus restricting the solution space.

A similar method was presented by Suzuki et al. [54], where handling of over-constrained and under-constrained problems is given special consideration.

Verroust et al. [56] have used the same theoretical foundations as Sunde [53]. They added some rules to extent the class of problems solved.

Rule-constructive approach provide a qualitative study of geometric constraints. Although it is a good approach for prototyping and experimentation, the extensive computations involved in the exhaustive searching and matching make it inappropriate for real world applications.

IV. GRAPH-BASED APPROACH

A. Graph representation

In graph-based methods, an undirected graph $G=(V,E)$ where $|V|=n$ and $|E|=m$ represents the constraint problem. The geometric elements are represented by the graph nodes and the constraints are the graph edges. The class of configurations solved by these methods is typically the ruler and compass constructive problems. In Computer-Aided Design, the graph-based approach has become dominant.

Example:

A dimensioning scheme defining a constraint problem is shown in figure 5. It involves six points and six lines. The constraints are six point-point distances, three line-line angles, twelve point-line implicit distances that are zero. The corresponding constraint graph is shown in figure 6. The edges are labeled with the values of the distance and angle dimensions. The unlabeled edges correspond to the implicit point-line distances.

B. Graph structure analysis

Graph-constructive solvers are stemming from graph theory. They are based on an analysis of the structure of the constraint graph. The graph constructive approach provides means for developing sound and efficient algorithms. Let us now give some definitions concerning the structural properties of the constraint graph when the degree of freedom of each geometric object is two and each constraint 'fixes' one degree of freedom.

Definition 1

A constraint graph $G=(V,E)$ where $|V|=n$ and $|E|=m$ is structurally well-constrained if and only if $m=2*n-3$ and $m'\leq 2*n'-3$ for any induced sub-graph $G'=(V',E')$ where $|V'|=n'$ and $|E'|=m'$ (see [33]).

Definition 2

A constraint graph $G=(V,E)$ contains a structurally over-constrained part if there is an induced sub-graph $G'=(V',E')$ having more than $2*n'-3$ edges.

Definition 3

A constraint graph $G=(V,E)$ is structurally under-constrained if it is not over-constrained and the number of edges is less than $2*n-3$. 
We must mention that the Laman's theorem [33] is proved only for point to point distance constraints and that the extension of his definitions may imply incorrect cases. A typical example of such cases is given by a triangle constrained with three angle constrainst. In figure 7.left, a triangle is constrained with three angle constraints (with $\alpha+\beta+\gamma=180^\circ$). The triangle is geometrically under-constrained but the constraint graph related to (figure 7.right) is well constrained.

The graph constructive approach uses only the structure of the constraint graph and forgets numerical information. A constraint graph can be structurally well-constrained but numerically under-constrained. The dimensioned scheme shown in figure 8 (left) is numerically under-constrained (the edges are labeled with distance arguments). Its corresponding constraint graph shown in figure 8 (right) is structurally well-constrained. Nonetheless, with the graph constructive approach, such cases will be detected during the construction phase.

C. Resolution principle

Graph-based algorithms for solving geometric constraint problems have two phases: an analysis phase and a construction phase. These algorithms are also called decomposition recombination methods.

During the first phase the graph of constraints is analyzed and decomposed in small sub-problems. Sequences of construction steps are derived. During the second phase the recombination is carried out and the construction steps are processed to place the geometric elements. Figure 9 shows an example of decomposition of a constrained scheme.

The placement steps correspond to standardized geometric construction steps (place a point at given distances from two points, place a line at prescribed angle from another line through a point, rotate and translate a sub-structure, ...).

D. Graph based solvers

In the early nineties, Owen [46] was the first to give an efficient and sound graph-based algorithm. This algorithm is quadratic and performs well on ruler and compass constructive problems. Following that, based on Owen's work, was developed a commercial system named D-Cubed. Since that and until recently, a lot of researches were conducted in this promising way. Among these works we can mention those done by Hoffmann's "constraint team" ([9], [17], [42], [43], [44], [25], [26], [19]), Lamure et al. [35], Lee et al. [38], Ait-Aoudia et al. [1], Jermann et al. [29], Gao et al. [20], Lee et al. [39], Sitharam et al [52], Gao et al. [21], Zhang et al. [57], Joan-Arinyo et al. [30], Podgorelec et al. [48].
V. Conclusion

We have classified the geometric constraint solvers in three broad categories, we summarise hereafter some advantages and drawbacks for each approach.

Numerical methods are $O(n^3)$ or worse. These methods suffer from the lack of 'geometric explanation' during the resolution process. Also, most numerical methods have difficulties for handling over-constrained or under-constrained schemes. The advantage of these methods is that they have the potential to solve large nonlinear system that may not be solvable using any of the other methods. Almost all existing solvers switch to algebraic methods when the given configuration is not solvable by the native method. Algebraic methods handle easily the 3D case. Speeding up the resolution must be considered further.

Symbolic methods are typically exponential in time and space. They can be used only for small systems. According to Lazard [36], computing the Gröbner bases of an irreducible system of degree two in ten unknowns is a hopeless case.

Rule-based solvers rely on the predicates formulation. Although they provide a qualitative study of geometric constraints, the "huge" amount of computations needed (exhaustive searching and matching) make them inappropriate for real world applications.

Graph-based methods can be very efficient. In Computer-Aided Design, the graph-based approach has become dominant. However, these methods are only applicable to particular kinds of problems, typically ruler and compass constructive problems. The graph-based approach to handle efficiently the 3D case deserves further studies. The theoretical framework is no longer applicable. The extension of Laman theory (section IV.B) to the 3D case will give: a constraint graph $G=(V,E)$ where $|V|=n$ and $|E|=m$ is structurally well-constrained if and only if $m=3n-6$ and $m' \leq 3n'-6$ for any induced sub-graph $G'=(V',E')$ where $|V'|=n'$ and $|E'|=m'$. The 3D constrained design shown in figure 11 involves eight points and eighteen point to point distances. This well known example verifies the formula given above but it is not structurally well constrained. The two sub-parts defined by the points $\{A,B,C,D,E\}$ and $\{A,G,H,F,E\}$can rotate around the $AE$ axis.

VI. References

[18] X.S. Gao and S.C. Chou. Solving geometric constraint systems : a symbolic approach and decision of Re-


