A Branch-and-Prune Algorithm for Solving Systems of Distance Constraints

Josep Maria Porta†, Federico Thomas‡, Lluís Ros‡ and Carme Torras‡

† IAS, Intelligent Autonomous Systems, Informatics Institute. Univ. of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands
porta@science.uva.nl
‡ Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Llorens Artigas 4-6, 2a planta, 08028 Barcelona, Catalonia, Spain
{fthomas,llros,ctorras}@iri.upc.es

***Abstract*— Given a set of affine varieties in $\Re^3$, i.e. planes, lines, and points, the problem tackled in this paper is that of finding all possible configurations for these varieties that satisfy a set of pairwise euclidean distances between them. Many problems in Robotics –such as the forward kinematics of parallel manipulators or the contact formation problem between polyhedral models– can be formulated in this way. We propose herein a strategy that consists in finding some distances, that are unknown a priori, and whose derivation permits solving the problem rather trivially. Finding these distances relies on a branch-and-prune technique that iteratively eliminates from the space of distances entire regions which cannot contain any solution. This elimination is accomplished by applying redundant necessary conditions derived from the Theory of Cayley-Menger determinants. The experimental results obtained qualify this approach as a promising one.**

I. INTRODUCTION

The resolution of systems of geometric constraints has aroused interest in many areas of Robotics (contact analysis, assembly planning, forward kinematics of parallel manipulators, path planning of closed-loop kinematic chains, etc.) and CAD/CAM (constraint-based sketching and design, interactive placement of objects, etc.). The solution of such problems entails finding object positions and orientations that satisfy all established constraints simultaneously.

Several methods are available for translating a system of geometric constraints into a set of algebraic equations to be solved. Thus, the general methods developed for finding all the roots of such sets of equations can be readily applied to this problem. Among all possible alternatives, our group has been exploring the interval-based approaches for one main reason: they are fully numerical, as opposed to those based on elimination theory or computer algebra. We first applied the Hansen algorithm –which uses an extension of the Newton method to interval arithmetic, known as interval Newton method– in conjuntion with some necessary conditions, that can be directly drawn from the problem itself, to speed up the convergence [3]. Afterwards, we applied the subdivision property of Bernstein polynomials which, while maintaining the quadratic convergence to the solutions of the Hansen algorithm, avoids the computation of derivatives [2].

This latter technique boils down to a remarkably simple algorithm when the problem can be described only by multilinear equations. Since the description of any arbitrary geometric constraint problem can be expressed in terms of such a set of equations plus a certain number of circle equations, we explored the application of only one of these equations at a time to reduce the search space [14]. This approach combined the success of the iterative application of necessary conditions and the simplicity of the multilinear equations. It also showed that the application of redundant necessary conditions permits delivering fairy small regions of the search space containing all the solutions without relying on a global consistency test.

We present here a step further in this progression, where we depart strongly from the usual formulation in that our variables are now *distances* instead of degrees of freedom linked to artificial reference frames. We still take advantage of the application of redundant sets of necessary conditions expressed as multilinear equations, but these conditions are now standardized coordinate-free equations derived from the Theory of Cayley-Menger determinants. The result is a branch-and-prune technique that obtains some distances unknown in the original problem, compatible with the established geometric constraints, which permit solving it rather trivially.

The paper is structured as follows. In Section II, Cayley-Menger determinants are briefly introduced. Using them, in Section III it is shown how geometric constraints, such as aligment or orthogonality, can be translated into constraints involving only distances. Then, the proposed branch-and-prune algorithm is detailed in Section IV. Two applications of the method in the areas of robot kinematics and geometric design are presented in Section V, and finally some conclusions are drawn in the closing section.

II. CAYLEY-MENGER DETERMINANTS

Let us define the function

$$\Xi(\mathbf{p}_1,\ldots,\mathbf{p}_n) = \begin{vmatrix} 0 & r_{12} & r_{13} & \cdots & r_{1n} & 1 \\ r_{21} & 0 & r_{23} & \cdots & r_{2n} & 1 \\ r_{31} & r_{32} & 0 & \cdots & r_{3n} & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & r_{n3} & \cdots & 0 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 0 \end{vmatrix},$$

where $\mathbf{p}_1,\ldots,\mathbf{p}_n$ are n points in $\Re^3$ and $r_{ij} = ||\mathbf{p}_i - \mathbf{p}_j||^2$, i.e. the square distance between $\mathbf{p}_i$ and $\mathbf{p}_j$. Obviously, $r_{ij} = r_{ji}$. The previous determinant is the general form of the Cayley-Menger determinant. It was first used by A. Cayley in 1841 [4], but it was not systematically studied until 1928, when K. Menger showed how it could be used to study convexity and other basic geometric problems [11]. Nowadays, this determinant plays a fundamental role in the so-called "Distance Geometry," a term coined by L. Blumenthal in [1] which refers to the analytical study of the Euclidean geometry in terms of invariants without resorting to artificial coordinate systems.

If $n = 2$,

$$\Xi(\mathbf{p}_1,\mathbf{p}_2) = 2\, r_{12}. \tag{1}$$

If $n = 3$,

$$\Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3) = -16\, A^2, \tag{2}$$

where A is the area of the triangle defined by $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. Actually, Eq. (2) is Herron's formula, which permits to obtain the area of a triangle in terms of the lengths of its edges.

If $n = 4$,

$$\Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4) = 288\, V^2, \tag{3}$$

where V is the volume of the tetrahedron defined by $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$. If $\Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4)$ vanishes, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ lie on the same plane. If it gives a negative value, the tetrahedron cannot be assembled with the given distances. Actually, Eq. (3) is known as Euler's tetrahedron formula.

If $n > 4$,

$$\Xi(\mathbf{p}_1,\ldots,\mathbf{p}_n) = 0 \tag{4}$$

because this determinant essentially gives the volume of a simplex in $\Re^{n-1}$ but, since this simplex is degenerate in $\Re^3$, its volume is zero. Note that equations of this type constitute necessary conditions that a set of interpoint distances must fulfill, if the point configuration they describe must be realizable in $\Re^3$.

III. GEOMETRIC CONSTRAINTS AS DISTANCE CONSTRAINTS

Many geometric constraints can be expressed in terms of distances (i.e., in a coordinate-free form) by using Cayley-Menger determinants. Below, we derive three such constraints: collinear points, orthogonal segments and point-line distance.

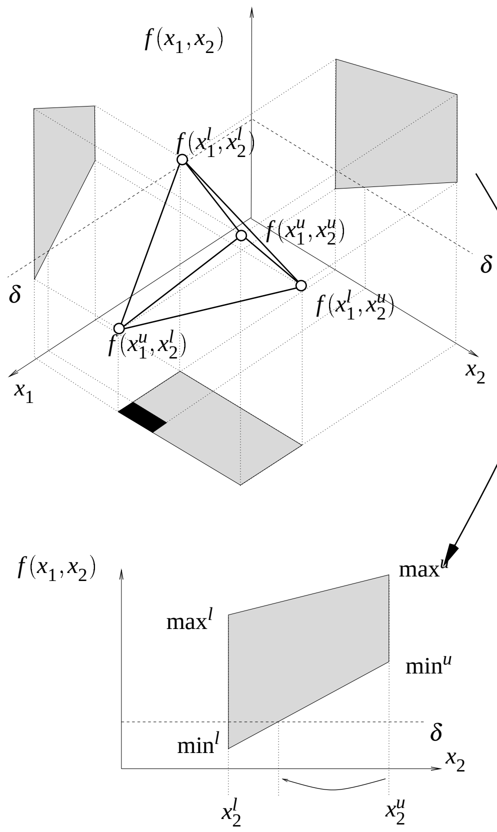


Fig. 1. Segment-trapezoid clipping.

Three points $\mathbf{p}_1,\mathbf{p}_2$, and $\mathbf{p}_3$ are *collinear* if, and only if, $\Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3) = 0$. This follows from Eq. (2), since the area of the triangle defined by three collinear points is null.

Two adjacent segments $\mathbf{p}_1\mathbf{p}_2$ and $\mathbf{p}_2\mathbf{p}_3$ are *orthogonal* if, and only if, $\Xi(\mathbf{p}_1,\mathbf{p}_2) + \Xi(\mathbf{p}_2,\mathbf{p}_3) - \Xi(\mathbf{p}_1,\mathbf{p}_3) = 0$. This is a rewriting of Pythagoras' theorem by using Eq. (1).

Finally, the distance d between a point $\mathbf{p}_1$ and a line passing through $\mathbf{p}_2$ and $\mathbf{p}_3$ satisfies the equation

$$\Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3) + 4 r_{23} d^2 = 0, \tag{5}$$

which follows from Eq. (2) and the fact that, in this case, $A^2 = r_{23} d^2/4$.

Section V will exemplify how kinematic or geometric constraint solving problems can be formulated and solved on the basis of distance constraints like those introduced above.

IV. THE ALGORITHM

The algorithm we present, based on that introduced in [14], solves a system of multilinear equations with n variables isolating the solutions contained in an initial box $\mathscr{B} \subset \Re^n$, by iteratively cutting off portions of $\mathscr{B}$ containing no root. Both Cayley-Menger determinants and identity relations $r_{ij} = r_{ji}$ are multilinear and, thus, this algorihtm can be used to solve systems of Cayley-Menger equations.

```
The Solver
  Input: A set of multilinear equations
  Output: A set of solution boxes ($S$)
  Process:
    $S \leftarrow \emptyset$
    $L \leftarrow$ Initial list of boxes
    while not empty(L)
        $\mathscr{B} \leftarrow$ first box(L)
        do
            $s \leftarrow$size($\mathscr{B}$)
            Reduce_Box($\mathscr{B}$)
        until empty($\mathscr{B}$) or size($\mathscr{B}$)$\leq \sigma$ or size($\mathscr{B}$)$/s > \rho$
        if not empty($\mathscr{B}$) then
          if size($\mathscr{B}$)$\leq \sigma$ then
            $S \leftarrow S \cup \{\mathscr{B}\}$
          else
            Split $\mathscr{B}$ into two sub-boxes: $\mathscr{B}_1, \mathscr{B}_2$
            Add $\mathscr{B}_1$ and $\mathscr{B}_2$ to L
          endif
        endif
    endwhile
```

Fig. 2. The main loop of the equation solver.

```
Reduce_Box($\mathscr{B}$)
  Input: A box defined as a set of intervals:
        $\mathscr{B} = \{[x_1^l, x_1^u], \dots, [x_n^l, x_n^u]\}$
  Output: The same box, but eventually resized
  Process:
    for each equation $f$ of the form $f(\mathbf{x}) = \delta$
      $V \leftarrow \{v_0, \dots, v_k\}$ (Indices of variables in $f$)
      $F = \{f(\mathbf{x}) \,|\, \mathbf{x} \in \{x_{v_0}^l, x_{v_0}^u\} \times \dots \times \{x_{v_k}^l, x_{v_k}^u\}\}$
      for each $v \in V$
        $\min^l \leftarrow \min\{f \mid f \in F, x_v = x_v^l\}$
        $\max^l \leftarrow \max\{f \mid f \in F, x_v = x_v^l\}$
        $\min^u \leftarrow \min\{f \mid f \in F, x_v = x_v^u\}$
        $\max^u \leftarrow \max\{f \mid f \in F, x_v = x_v^u\}$
        Trapezoid_Clipping($x_v^l, x_v^u$,
                                $\min^l, \max^l, \min^u, \max^u, \delta$)
      endfor
    endfor
```

Fig. 3. The **Reduce_Box** function. $f(x_0, \dots, x_k)$ refers the evaluation of the equation f at point $(x_0, \dots, x_k)$.

The reduction of a given box is based on the following lemma, which is a direct consequence of Theorem 1.1 in [16]: The point $(\mathbf{x}, f(\mathbf{x})) \in \Re^{n+1}$, where f is a scalar multilinear function and $\mathbf{x} = (x_1, \dots, x_n) \in [x_1^l, x_1^u] \times \cdots \times [x_n^l, x_n^u]$, is contained in the convex hull of the 2^n points $\{(\mathbf{x}, f(\mathbf{x})) | x_k \in \{x_k^l, x_k^u\}\}$.

Assume we want to find all solutions of a multilinear equation $f(\mathbf{x}) = \delta$, for $\mathbf{x} = (x_1, x_2)$ in the box $\mathscr{B} = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \in \Re^2$. Since $(\mathbf{x}, f(\mathbf{x}))$ must lie within the convex hull of the 2^2 points $\{(\mathbf{x}, f(\mathbf{x}))| \ \mathbf{x} \in \{x_1^l, x_1^u\} \times \{x_2^l, x_2^u\}\}$, we can compute the convex hull of these points in $\Re^3$ and

	configuration a		configuration b	
	non redundant	redundant	non redundant	redundant
t	< 0.01 (0.09)	< 0.01 (0.35)	0.03 (0.28)	0.03 (2.15)
b_p	7 (851)	3 (1477)	65 (2173)	39 (6841)
n_s	2 (86)	2 (67)	27 (680)	8 (764)

TABLE I
THE ALGORITHM'S PERFORMANCE ON THE FORWARD KINEMATICS OF THE OCTAHEDRAL MANIPULATOR.

intersect it with the plane $f(\mathbf{x}) = \delta$ to obtain a polygon whose rectangular hull gives a better bound for the solutions. Although this method is inefficient for a high number of variables, it can be simplified through the following variation: we simply project the hull onto each coordinate plane, as depicted in Fig. 1 (top), and intersect each of the resulting trapezoids with the $f(\mathbf{x}) = \delta$ line, as shown in Fig. 1 (bottom). Usually, these segment-trapezoid clippings reduce the ranges of some variables giving a smaller box (the black rectangle in Fig. 1-top) but still bounding the root locations. The experiments show that, although this strategy produces less pruning than the convex hull-plane clipping, it results advantageous due to the lower cost of its operations.

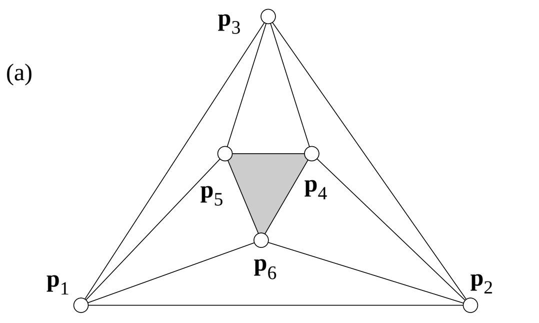
Our solver (Figs. 2 and 3) reduces the boxes that bound the initial search space by applying the trapezoid-line clipping just described. If, for a given box, there is no intersection between the line and the trapezoid, the box contains no solution and we can simply stop the exploration in the search space delimited by that box. After the application of the clipping process for all equations and variables, boxes whose longest side becomes smaller than a given threshold are considered solution boxes. In Fig. 2, this longest side and the threshold are denoted by *size*$(\mathscr{B})$ and σ, respectively. Finally, boxes that cannot be significantly reduced (i.e., the reduction ratio of their longest side is above a given threshold ρ) are split and the two sub-boxes are added to the list of boxes still to be processed. The result of this process is a set of small boxes S that includes all solutions for a given set of equations.

V. EXPERIMENTS

The previous algorithm has been implemented in C and we next show how it performs in two test cases: solving the forward kinematics of octahedral manipulators, and finding all lines simultaneously tangent to four spheres.

A. Solving the octahedral manipulator

An octahedral manipulator is formed by two triangles, the *base* $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ and the *platform* $\mathbf{p}_4\mathbf{p}_5\mathbf{p}_6$, joined by six linearly-actuated *legs*: $\mathbf{p}_1\mathbf{p}_5$, $\mathbf{p}_1\mathbf{p}_6$, $\mathbf{p}_2\mathbf{p}_6$, $\mathbf{p}_2\mathbf{p}_4$, $\mathbf{p}_3\mathbf{p}_4$ and $\mathbf{p}_3\mathbf{p}_5$ (Fig. 4a). The forward kinematics problem is to find all poses of the platform (relative to the base) that are



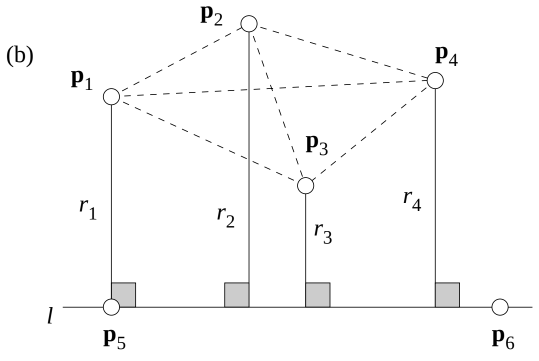


Fig. 4. Points and parameters involved in the test cases.

compatible with the six specified leg-lengths. No closed-form solution is known to this problem, but numerical procedures have been given that involve finding the roots of an 8th-degree univariate polynomial, obtained by symbolic elimination techniques [5], [13]. Using Cayley-Menger determinants, though, it is possible to give the following simple formulation of the problem, directly solvable by the above algorithm. To this end, consider the following Cayley-Menger equations:

$$\begin{aligned}\Xi(\mathbf{p}_1,\mathbf{p}_3,\mathbf{p}_4,\mathbf{p}_5,\mathbf{p}_6)=0,\\ \Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4,\mathbf{p}_6)=0.\end{aligned} \tag{6}$$

Note that, among all involved distances, only $r_{1,4}$ and $r_{3,6}$ are unknown, and that once the system is solved for them, we can determine the spatial position of the three points of the platform by trilateration [17], since each of these points will have a tripod of known lengths with three points at a known position. We may now use our algorithm to solve these equations. Figs. 5a and 5b show the results for two different sets of leg-lengths. In both cases, the base and platform triangles are equilateral, of side $sqrt(3)$ and $sqrt(3)/2$, respectively. Fig. 5a shows the solution boxes found when the leg-lengths are set to $r_{1,5}=r_{2,6}=r_{3,4}=4.25$ and $r_{1,6}=r_{2,4}=r_{3,5}=5.75$, a case hereafter referred to as configuration "a". Fig. 5b shows the solution boxes when all leg-lengths are set to 4.75, a case hereafter referred to as configuration "b". Insight into the behaviour of the solver may be get by comparing these two outputs with

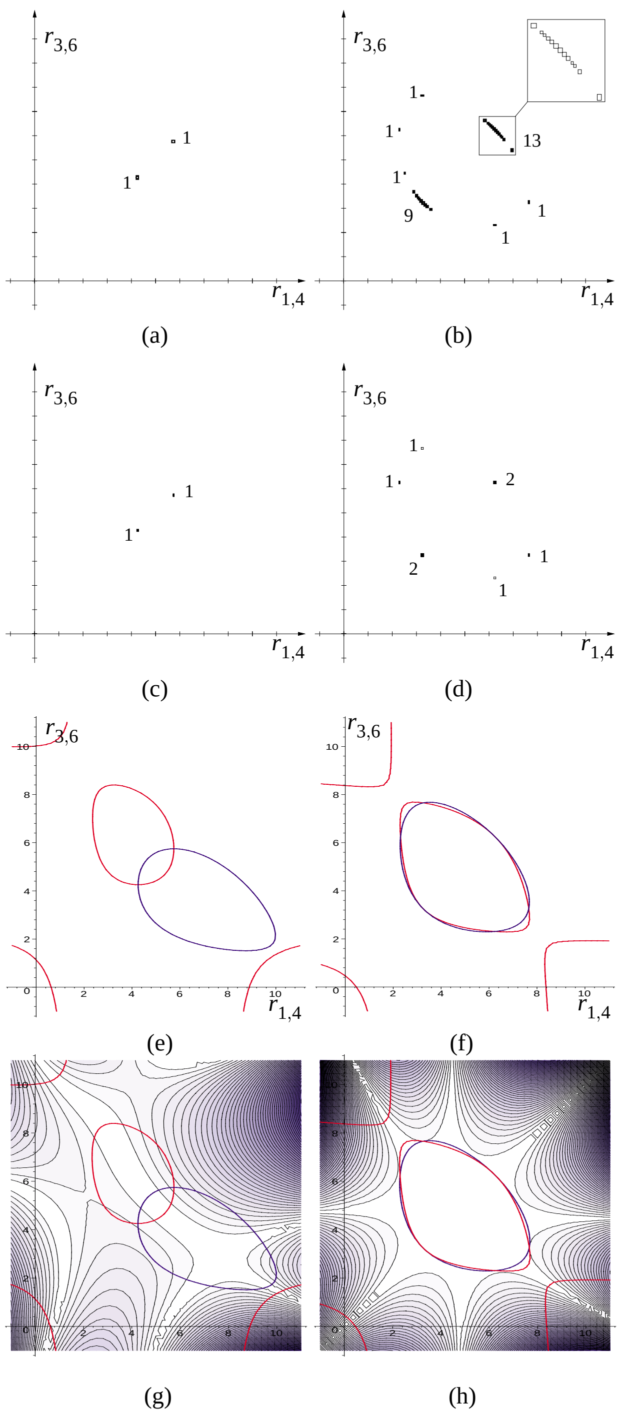


Fig. 5. Solving the octahedral manipulator. The numbers in (a), (b), (c) and (d) indicate the amount of solution boxes returned around each solution point.

the corresponding plots of the implicit curves of Eqs. (6), shown in Figs. 5e and 5f. Note that while in configuration "a" the two curves are rather different and intersect only in two points, in configuration "b" they are quite close to one another and intersect in six points, with tangency on two of them. This proximity explains why our solver gives larger clusters of boxes in Fig. 5b than in Fig. 5a.

We may add redundant equations to the system of Eqs. (6). For example, if we add the remaining Cayley-Menger equations of five points,

$$\begin{aligned}\Xi(\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4,\mathbf{p}_5,\mathbf{p}_6)=0,\\ \Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_4,\mathbf{p}_5,\mathbf{p}_6)=0,\\ \Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_5,\mathbf{p}_6)=0,\\ \Xi(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4,\mathbf{p}_5)=0,\end{aligned} \tag{7}$$

we end up with a system of six equations in three unknowns. The solution boxes found by the solver are displayed in Figs. 5c and 5d, for configurations "a" and "b", respectively. Comparing Figs. 5a and 5b with Figs. 5c and 5d, we clearly see that the use of redundant equations produces extra pruning, and that the solutions are bounded with higher accuracy. Table I shows the execution time t for both configurations[1], the number b_p of boxes processed by the algorithm, and the number n_s of solution boxes found. These statistics are separately given for the non-redundant formulation of Eqs. (6), and for the redundant one involving Eqs. (6) and (7). In parentheses, the table also gives t, b_p and n_s, for a slightly modified version of the algorithm that uses interval arithmetic to compute the vertical sides of the trapezoids, instead of evaluating the 2^n control points involved. In all cases, the global control parameters have been set to $\sigma = 0.1$ and $\rho = 0.9$. The table clearly shows the positive effect of adding redundant equations: although the two configurations are solved in practically the same time, in the redundant case fewer boxes have to be explored. It is remarkable that, for configuration "a", the redundancy of equations allows to isolate the solutions by only exploring three boxes, the minimum required when two solutions exist. In configuration "b", the solver also isolates the solutions, but returns whole clusters of boxes for those lying in tangency points (Fig. 5b). This effect is nevertheless reduced when adding redundancy, as the delivered clusters contain only two boxes each, as shown in Fig. 5d.

The cost of processing each box during the segment-trapezoid clipping is $O(2^n)$, where n is the maximum number of variables per equation. When using interval arithmetic in this process, this cost is reduced to $O(n)$ but, despite this lower complexity, we observe that both t and b_p increase considerably in this case, as shown in the table. This is due to the fact that interval arithmetic yields looser bounds for the vertical sides of the trapezoids.

Finally, this example is useful to illustrate how the presented algorithm does not suffer from two common problems of classical root-finding procedures. On the one hand, it is immune to singularities of the Jacobian of the equations because it does not use derivative information. Certainly, this is a typical drawback of Newton-Raphson methods. Given a system of equations $F(x) = 0$, such methods iteratively work on an estimation x_i of the solution to derive a better estimation x_{i+1} using the recurrence $x_{i+1} = x_i - D^{-1} \cdot F(x_i)$, where D is the matrix of partial derivatives of $F(x)$. Clearly, when D is close to singular, the method may fail to converge. Figs. 5g and 5h show iso-contours of the determinant of D for configurations "a" and "b", overlaid with the curves in Figs. 5e and 5f, respectively. The white areas correspond to points where this determinant is lower than 10^{-8}. One can verify that, using the Newton-Raphson routines of MAPLE, for example, it is impossible to compute the two solutions where the curves in Fig. 5f are tangent, precisely because they lie inside a region of near-singularitiness of D. On the other hand, it is well-known that the numerical stability of polynomial root finding is often surprisingly low [15], [7] and that a very small perturbation in just a few coefficients can yield solutions completely different from the intended ones. Classic solutions to the forward kinematics problem that rely on solving a resultant polynomial must carefully deal with this issue, specially in configurations of the platform near a singularity, where solutions may completely be lost. Contrarily, our algorithm is robust in this sense because it directly works with the input equations.

[1]Time in seconds, on a Pentium IV PC at 1.8 GHz.

	$\sigma = 0.1$	$\sigma = 0.01$
t (sec.)	315	385
b_p	7875	14579
n_s	870	813

TABLE II
THE ALGORITHM'S PERFORMANCE WHEN COMPUTING ALL LINES TANGENT TO FOUR SPHERES.

B. All lines tangent to four spheres

Given four spheres of radii $r_1,\ldots,r_4$ in $\Re^3$, with their centers located in $\mathbf{p}_1,\ldots,\mathbf{p}_4$, we want to find their common tangent lines. Equivalently, the problem can be stated as finding all possible lines that keep the prescribed distances $r_1,\ldots,r_4$ to the four points $\mathbf{p}_1,\ldots,\mathbf{p}_4$. This problem was first formulated by Larman [9], and later discussed by Durand [6], Karger [8], and Verschelde [18], and finds several applications in Computer Graphics and Computational Geometry. It has been proven that there are at most twelve discrete solutions and that this bound is tight. A method to find them has been recently given by MacDonald et al. [10] who formulate it as a system of two algebraic equations, a cubic and a quartic, in the involved point and vector coordinates. After elimination, this yields a seventh degree univariate resultant that must be numerically solved. As an alternative, one can arrive at the following coordinate-free formulation, directly tackleable with the presented

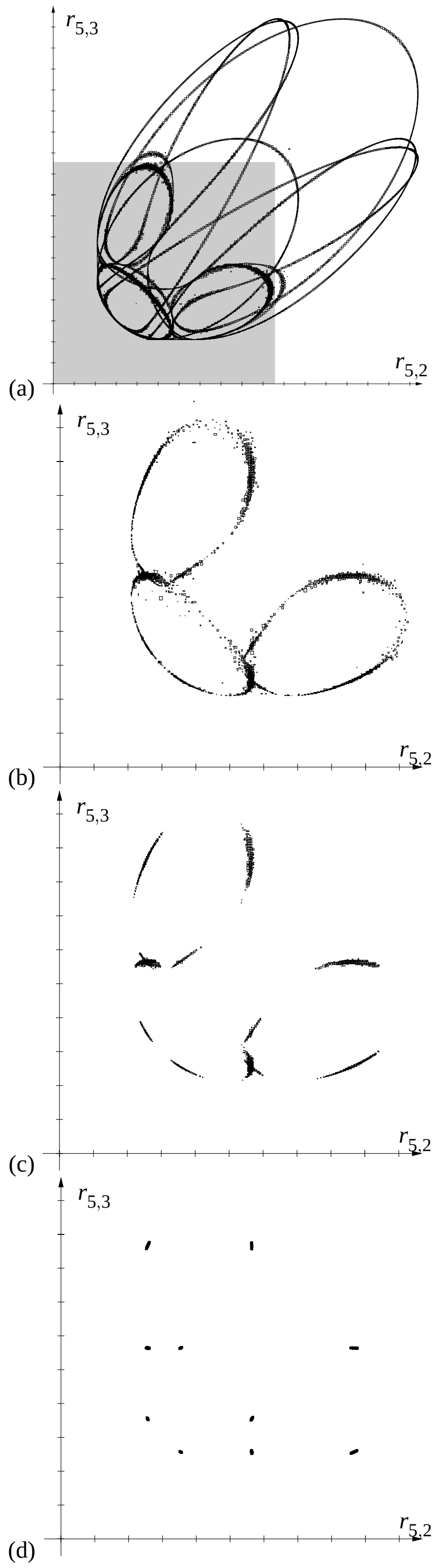


Fig. 6. Output for the second test case. The lengths of the axes in (b), (c) and (d) correspond to the shaded area in (a).

constraint solver.

First, we characterize the tangent line l by two points on it, say $\mathbf{p}_5$ and $\mathbf{p}_6$, placed a unit distance apart, such that $\mathbf{p}_5$ is the tangent point of l with the first sphere. (See Fig. 4b.) With this, and using the right triangle $\mathbf{p}_1\mathbf{p}_5\mathbf{p}_6$, we directly see that $r_{1,6} = r_1 + 1$. Finally, we state the following distance constraints among all points in $S = \{\mathbf{p}_1, \ldots, \mathbf{p}_6\}$, defining a redundant system of ten equations in six unknowns:

C1: Three constraints of the form of Eq. (5) to force that the distance from each of $\mathbf{p}_2, \ldots, \mathbf{p}_4$ to l be $r_2, \ldots, r_4$, respectively.

C2: The two Cayley-Menger equations of five points $\Xi(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5) = 0$, and $\Xi(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_6) = 0$, each involving three unknowns.

C3: The remaining four Cayley-Menger equations of five points of S, three involving four unknowns, and one involving six unknowns.

C4: The unique Cayley-Menger equation of the six points in S, with six unknowns.

One can now use the solver to treat them all together, but it is illustrative to successively apply it to larger subsets of these equations instead, and see the outputs. Let us study the case where all inter-center distances are $\sqrt{8}$, and all radii are 1.45, for which 12 solutions exist [10]. If we start by setting $\sigma = 0.1$ and we just consider the five constraints in **C1** and **C2**, we obtain the one-dimensional continuum of solutions depicted in Fig. 6a. The continuum disappears when we solve **C1**, **C2** and **C3** together, as seen in Fig. 6b, giving rise to very large clusters of solution boxes. These can be further reduced if the last constraint **C4** is taken into account, to get the box clusters in Fig. 6c. At this point we have exhausted all possible distance constraints between the selected points and we cannot further reduce the clusters, unless we ask a higher accuracy to the solver. If we do this, by setting $\sigma = 0.01$, we get the small clusters in Fig. 6d, each corresponding to one of the 12 solutions of the problem. (Actually, two pairs of clusters appear overlaid, but they can be seen separated by choosing a proper projection.)

Table II gives the values of t, b_p and n_s for the last two experiments. The $\sigma = 0.1$ and $\sigma = 0.01$ columns correspond, respectively, to the computation of Figs. 6c and 6d. Both experiments have been done with $\rho = 0.99$. We note that the time to compute the solutions does not increase substantially, despite the fact that σ has been decreased by one order of magnitude in the second experiment. Furthermore, although a higher number of boxes is processed for $\sigma = 0.01$, the final number of solution boxes remains practically the same as for $\sigma = 0.1$. This is not casual. One can see that, from a certain point, after asking higher and higher accuracies to the solver, the number of boxes around each solution point will practically remain constant. This phenomenon, known as the *cluster effect*, was already observed in [12], and its avoidance constitutes part of our current research.

VI. CONCLUSIONS

We have presented a general algorithm for solving systems of geometric constraints. The algorithm is complete in the sense that it does not lose any solution. The combination of a branch-and-prune technique with the use of coordinate-free standardized constraints has proven effective to achieve this.

According to our experiments, the addition of redundant constraints speeds up, in general, the resolution process and reduces the number of final boxes delivered. Although not illustrated by the presented examples, the addition of redundant variables, on the contrary, usually introduces a trade-off between the number of final boxes and execution times: as the number of redundant variables is increased, the solver needs longer execution times but, in return, it obtains a lower number of final boxes.

The algorithm as it stands leaves many choices open, as it is usually the case in constraint-based search (variable ordering, constraint selection, redundacy dosage, etc.). This offers a range of possibilities to speed up the resolution process, which we will tackle in future research by devising good heuristics for the choice points above.

VII. REFERENCES

[1] L.M. Blumenthal, *Theory and Applications of Distance Geometry,* Oxford University Press, 1953.

[2] C. Bombín, L. Ros, and F. Thomas, "A Concise Bézier Clipping Technique for Solving Inverse Kinematics Problems," in *Advances in Robot Kinematics*, J. Lenarcic and M.M. Stanisic (Eds.), pp. 53-61, Kluwer Academic Publishers, 2000.

[3] A. Castellet and F. Thomas, "An Algorithm for the Solution of Inverse Kinematic Problems Based on an Interval Method," in *Advances in Robot Kinematics*, M. Husty and J. Lenarcic (Eds.), pp. 393-403, Kluwer Academic Publishers, 1998.

[4] A. Cayley, "A Theorem in the Geometry of Position," *Cambridge Mathematical Journal*, Vol. II, pp. 267-271, 1841.

[5] M. Griffis and J. Duffy, "A Forward Displacement Analysis of a Class of Stewart Platforms," *Journal of Robotic Systems,* Vol. 6, No. 6, pp. 703-720, 1989.

[6] C. Durand, *Symbolic and Numerical Techniques for Constraint Solving,* PhD. Thesis, Purdue University, 1998.

[7] D. Kahaner, C. Moler, S. Nash, *Numerical Methods and Software*, Prentice Hall, 1989.

[8] A. Karger, "Classical Geometry and Computers," *J. Geom. Graph.*, Vol. 2, pp. 7-15, 1998.

[9] D. Larman, Problem posed in the problem session of the DIMACS workshop on arrangements, Rutgers University, New Brunswick, NJ, 1990.

[10] I.G. Macdonald, J. Pach, and T. Theobald, "Common Tangents to Four Unit Balls in $\Re^3$," *Discrete Computational Geometry*, Vol. 26, No. 1, pp. 1-17, 2001.

[11] K. Menger, "New Foundation for Euclidean Geometry," *American Journal of Mathematics,* No. 53, pp. 721-745, 1931.

[12] A. Morgan and V. Shapiro, "Box-Bisection for Solving Second-Degree Systems and the Problem of Clustering," *ACM Transactions on Mathematical Software,* Vol. 13, No. 2, pp. 152-167, 1987.

[13] P. Nanua, K.J. Waldron, and V. Murthy, "Direct Kinematic Solution of a Stewart Platform," *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 4, pp. 438-444, 1990.

[14] J.M. Porta, L. Ros, F. Thomas, C. Torras, "Solving Multi-Loop Linkages by Iterating 2D Clippings", in *Advances in Robot Kinematics*, F. Thomas and J. Lenarcic (Eds.), pp. 255-264, Kluwer Academic Publishers, 2002.

[15] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed., Mc Graw-Hill, 1978.

[16] A.D. Rikun, "A Convex Envelope Formula for Multilinear Functions," *J. of Global Optimization*, vol. 10, pp. 425-437, 1997.

[17] F. Thomas, E. Ottaviano, L. Ros and M. Ceccarelli, "Coordinate-Free Formulation of a 3-2-1 Wire-Based Tracking Device Based on Cayley-Menger Determinants," Also presented in this conference.

[18] J. Verschelde, "Polynomial Homotopies for Dense, Sparse and Determinantal Systems," Preprint #1999-041, Department of Mathematics, Michigan State University, 1999.