J.-P. Merlet

INRIA Sophia-Antipolis, France

Solving the Forward Kinematics of a Gough-Type Parallel Manipulator with Interval Analysis

Abstract

We consider in this paper a Gough-type parallel robot and we present an efficient algorithm based on interval analysis that allows us to solve the forward kinematics, i.e., to determine all the possible poses of the platform for given joint coordinates. This algorithm is numerically robust as numerical round-off errors are taken into account; the provided solutions are either exact in the sense that it will be possible to refine them up to an arbitrary accuracy or they are flagged only as a "possible" solution as either the numerical accuracy of the computation does not allow us to guarantee them or the robot is in a singular configuration. It allows us to take into account physical and technological constraints on the robot (for example, limited motion of the passive joints). Another advantage is that, assuming realistic constraints on the velocity of the robot, it is competitive in term of computation time with a real-time algorithm such as the Newton scheme, while being safer.

KEY WORDS—forward kinematics, parallel robot

1. Introduction

1.1. Robot Geometry

In this paper we consider a six-degrees-of-freedom (6-DOF) parallel manipulator (Figure 1) consisting of a fixed base plate and a mobile plate connected by six extensible links. Leg i is attached to the base with a ball-and-socket joint whose center is A_i while it is attached to the moving platform with a universal joint whose center is B_i. The length of the legs (the distance between A_i and B_i) will be denoted by ρ_i. A reference frame (A_1, x, y, z) is attached to the base and a mobile frame (B_1, x_r, y_r, z_r) is attached to the moving platform.

The International Journal of Robotics Research
Vol. 23, No. 3, March 2004, pp. 221-235,
DOI: 10.1177/0278364904039806

1.2. The Forward Kinematics Problem

The forward kinematics (FK) problem may be stated as: being given the six leg lengths, find the current pose S_c of the platform, i.e., the pose of the robot when the leg lengths have been measured.

Although it may seem that this problem has been addressed in numerous works, it has never been fully solved. Indeed, as we will see, all authors have addressed a somewhat different (although related) problem $\mathcal{P}$: being given the six leg lengths, find all the n possible poses $\mathcal{S} = \{S_1, \ldots, S_n\}$ of the platform. It may be accepted that solving $\mathcal{P}$ is the first step for solving the FK problem as soon as some method allows us to determine which solution S_j in the solution set of $\mathcal{P}$ is the current pose S_c of the robot. Unfortunately, no such method is known to date, even for planar parallel robots. This paper will also address the $\mathcal{P}$ problem, although we will be able to take into account, during the calculation, realistic constraints on the robot motion that may reduce the number of solutions.

Problem $\mathcal{P}$ is now considered as a classical problem in kinematics and is also used in other communities as a difficult benchmark. Raghavan (1991) and Ronga and Vust (1992) were the first to establish that there may be up to 40 complex and real solutions to $\mathcal{P}$ while Husty (1996) succeeded in providing a univariate polynomial of degree 40 that allows us to determine all the solutions. Dietmaier (1998) exhibited configurations for which there were 40 real solution poses.

1.3. Solving Method for the Forward Kinematics

The methods traditionally used to solve $\mathcal{P}$ may be classified as:

- the elimination method;
- the continuation method;
- the Gröebner basis method.

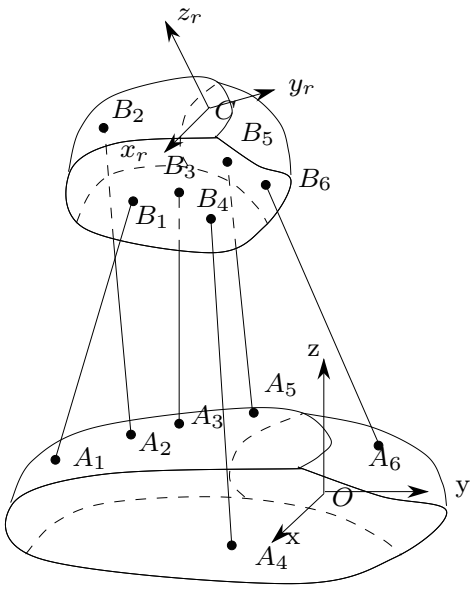

Fig. 1. Gough platform.

All these methods assume an algebraic formulation of the problem with n unknowns, $x_1, \ldots, x_n$. These methods will be described intuitively without trying to be rigorous.

In the elimination method (Innocenti 2001; Lee and Shim 2001a) each equation of the system is expressed as a linear equation in term of monomials $\prod x_1^{i_1} \ldots x_n^{i_n}$ (including the constant monomial 1) in which one of the unknowns, x_k, is supposed to be constant (i.e., the coefficients of the equations are functions of x_k). Additional equations are obtained by multiplying the initial equations by a monomial until we obtain a square system of linear equations that can be expressed in matrix form as

$$\mathbf{A}(\mathbf{x_k})\mathbf{X} = \mathbf{0} \tag{1}$$

where $\mathbf{X}$ is a set of monomials including the constant monomial 1. Due to this constant monomial, the above system has a solution only if $|\mathbf{A}(x_k)| = 0$, which is a univariate polynomial P_e in x_k. After solving this polynomial, a backtrack mechanism allows us to determine all the other unknowns for each root of the polynomial P_e. The main weakness of this method is the calculation of $|\mathbf{A}|$; usually $\mathbf{A}$ is a rather large matrix and its determinant cannot be calculated in closed form. Most authors propose to use a numerical method to evaluate the coefficients of the polynomial $|\mathbf{A}|$; the determinant (of order n), which is a linear function of the polynomial coefficients, is calculated numerically for $n + 1$ values of x_k and therefore the coefficients can be obtained by solving a system of $n + 1$ linear equations. However, such a procedure is numerically unstable and hence there is no guarantee of the correctness of the solutions.

An elimination method has been used by Husty (1996) to obtain a 40th-order polynomial but using only symbolic computation and a careful elimination process that guarantee that we obtain the exact polynomial coefficients; unfortunately, this procedure seems to be difficult to automate.

To solve a system of equations $\mathbf{F}(\mathbf{X}) = 0$, the continuation method (Raghavan 1991; Sreenivasan and Nanua 1992; Liu and Yang 1995; Wampler 1996) uses an auxiliary system $\mathbf{G}(\mathbf{X}) = \mathbf{F} + (1 - \lambda)(\mathbf{F_1} - \mathbf{F}) = 0$, where $\mathbf{F_1}$ is a system "similar" to $\mathbf{F}$, in the sense that it has at least the same number of solutions as $\mathbf{F}$, of which all the solutions are known and λ is a scalar. When λ is equal to 0, $\mathbf{G} = \mathbf{F_1}$ and consequently the solutions of $\mathbf{G}$ are known. These solutions are used as an initial guess to solve, using a Newton scheme, a new version of $\mathbf{G}$ obtained for $\lambda = \epsilon$ where ϵ has a small value. This process is repeated for $\lambda = 2\epsilon$ using the solutions of the previous run as an initial guess and so on until $\lambda = 1$ for which $\mathbf{G} = \mathbf{F}$. In other words, starting from a system with known solutions we follow the solution branches of a system that slowly evolves toward $\mathbf{F}$. The main weakness of this approach is that it is necessary to follow a large number of branches to find all the solutions of $\mathbf{F}$. In our case, $\mathbf{F_1}$ has to have at least 40 solutions and hence 40 branches will be followed, some of which will vanish if the FK problem has less than 40 solutions. Furthermore, numerical robustness is difficult to ensure if a singularity is encountered when following the branches.

In the Gröebner basis approach, the property is used that the solutions of any algebraic system $\mathbf{F}$ are also solutions of various other systems of equations in some other unknowns y_i. Among all these systems, one of them has the property of being triangular, i.e., the system has a first equation in one unknown y_1, the second equation has only y_1, y_2 as unknowns and so on, until the last equation with unknowns $y_1, \ldots, y_n$. Hence all the solutions of this system can be obtained by solving in sequence the first equation, then the second and so on. Such a triangular system can be obtained by using the Buchberger algorithm (Lazard 1992; Faugère and Lazard 1995). Although this method is currently the fastest to solve in a guaranteed manner, the FK problem (using the FGb and the RealSolving algorithms of Faugère[1] and Rouillier (1995, 2003)) this approach can only be used when the coefficients of the equations are rational (in which case the results are certified) and its implementation involves the use of large integers.

2. Solving with Interval Analysis

2.1. Interval Analysis

Interval analysis is an alternative numerical method that can be used to determine all the solutions to a system of equations and inequalities systems within a given search space.

1. See http://www-calfor.lip6.fr/~jcf/index.html.

An interval X is defined as the set of real numbers x verifying $\underline{x} \leq x \leq \overline{x}$. The "width" $w(X)$ of an interval X is the quantity $\overline{x} - \underline{x}$ while the "mid-point" $M(X)$ of the interval is $(\overline{x} + \underline{x})/2$. The "mignitude" ("magnitude") of an interval X is the smallest (highest) value of $|\underline{x}|, |\overline{x}|$. A "point interval" X is obtained if $\underline{x} = \overline{x}$. A "box" is a tuple of intervals and its width is defined as the largest width of its interval members, while its center is defined as the point whose coordinates are the mid-point of the ranges.

Let f be a real-valued function of n unknowns $\mathbf{X} = \{x_1, \ldots, x_n\}$. An interval evaluation F of f for given ranges $\{X_1, \ldots, X_n\}$ for the unknowns is an interval Y such that

$$\begin{aligned} &\forall \mathbf{X} = \{x_1, \ldots, x_n\} \in \mathcal{X} = \{X_1, \ldots, X_n\} \\ &\underline{Y} \leq f(\mathbf{X}) \leq \overline{Y}. \end{aligned} \tag{2}$$

In other words, $\underline{Y}$, $\overline{Y}$ are lower and upper bounds for the values of f when the unknowns are restricted to lie within the box $\mathcal{X}$.

There are numerous ways to calculate an interval evaluation of a function (Hansen 1992; Moore 1979). The simplest is the natural evaluation in which all the mathematical operators in f are substituted by their interval equivalent to obtain F. For example, the classical addition is substituted by an interval addition defined as

$$X_1 + X_2 = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}].$$

Interval equivalents exist for all the classical mathematical operators and hence interval arithmetics allows us to calculate an interval evaluation for most non-linear expressions, whether algebraic or not. For example, if $f(x) = x + \sin(x)$, then the interval evaluation of f for $x \in [1.1, 2]$ can be calculated as

$$\begin{aligned} F([1.1, 2]) = [1.1, 2] + \sin([1.1, 2]) = [1.1, 2] \\ + [0.8912, 1] = [1.9912, 3]. \end{aligned}$$

Interval evaluation exhibits interesting properties, as follows.

1. If $0 \notin F(\mathcal{X})$, then there is no value of the unknowns in the box $\mathcal{X}$ such that $f(\mathbf{X}) = 0$. In other words, the equation $f(\mathbf{X})$ has no root in the box $\mathcal{X}$.

2. The bounds of the interval evaluation F usually overestimate the minimum and maximum of the function over the box $\mathcal{X}$, but the bounds of F are exactly the minimum and maximum if there is only one occurrence of each unknown in f (Property A).

3. Interval arithmetics can be implemented taking into account round-off errors. For example, the interval evaluation of $f = x/3$ when X is the point interval [1,1] will be the interval $[\alpha_1, \alpha_2]$ where α_1, α_2 are the closest floating point numbers, respectively lower and greater than $0.3333\ldots$. There are numerous interval arithmetics packages implementing this property. One of the most well known is BIAS/Profil[2] using the C `double` for interval representation. However, a promising new package is MPFI (Revol and Rouillier 2002), based on the multi-precision software MPFR developed by the SPACES project[3], in which the interval is represented by a number with an arbitrary number of digits.

2.2. Basic Solving Algorithm

Interval analysis based algorithms have been used in robotics for solving the inverse kinematic of serial robots (Kiyoharu, Ohara, and Hiromasa 2001; Tagawa et al. 1999) and parallel robots FK (Castellet 1998; Didrit, Petitot, and Walter 1998; Jaulin et al. 2001), workspace analysis (Chablat, Wenger, and Merlet 2002; Merlet 1999), singularity detection (Merlet and Daney 2001), evaluating the reliability of parallel robots (Carreras et al. 1999), optimal placement of robots (Tagawa et al. 2001), mobile robot localization (Bouvet and Garcia 2001) and trajectory planning (Piazzi and Visioli 1997). However, interval analysis is a more complex method than may be thought at a first glance and we will present in this paper various improvements that have a drastic influence on the efficiency.

We start with the most basic solving algorithm that may be derived from the properties of interval arithmetics. Let $B_0 = \{X_1, \ldots, X_n\}$ be a box and $f = \{f_1(\mathbf{X}), \ldots, f_n(\mathbf{X})\}$ a set of equations to be solved within B_0. A list $\mathcal{L}$ will contain a set of boxes and initially $\mathcal{L} = \{B_0\}$. An index i, initialized to 0, will indicate which box B_i in $\mathcal{L}$ is currently being processed, while n will denote the number of boxes in the list. The interval evaluation of the function f_j for the box B_i will be denoted $F_j(B_i)$. A threshold ϵ will be used and, if the width of the interval evaluation of all the functions for a box B_i is lower than ϵ and includes 0, then B_i will be considered as a solution of the system. The algorithm proceed along the following steps.

1. If $i > n$, then return to the solution list.

2. If at least one $F_j(B_i)$ exists such that $0 \notin F_j(B_i)$, then $i = i + 1$ and go to 1.

3. If $\forall\ j \in [1, n]\ 0 \in F_j(B_i)$ and $w(F_j(B_i)) \leq \epsilon$, then store B_i in the solution list, $i = i + 1$ and go to 1.

4. Select the unknown k whose interval has the largest width in B_i. Bisect its interval in the middle point and create two new boxes from B_i as $\{X_1, \ldots, X_{k-1}, [\underline{X_k}, (\underline{X_k} + \overline{X_k})/2], \ldots, X_n]\}$ and $\{X_1, \ldots, X_{k-1}, [(\underline{X_k} + \overline{X_k})/2, \overline{X_k}], \ldots, X_n]\}$. Store these two boxes as B_{n+1}, B_{n+2}, $n = n + 2$, $i = i + 1$ and go to 1.

2. http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html.
3. http://www.mpfr.org.

Note that the storage method used here for the boxes is not very efficient as far as memory management is concerned. A first improvement is to substitute the box B_i by one of the two boxes that are created when bisecting it. A second improvement, denoted a depth first storage mode, is to store the second box at position $i + 1$ after a shift of the existing boxes. This ensures that the width of B_i is always decreasing until either the box is eliminated or a solution is found. In this mode, for a system of n equations in n unknowns, the width of B_i is at least divided by 2 after n bisection. If the width of the initial box B_0 is w_0 the number N of boxes that are needed is such that $2^{(K/n)} = w_0/\epsilon$ and hence $N = n\,log(w_0/\epsilon)/log(2)$. For example, if $n = 9$, $w_0 = 10$ and $\epsilon = 10^{-6}$, we obtain that the number of boxes of $\mathcal{L}$ should be 210 (to which we must add the memory to store the solutions). Hence, even with a high accuracy for the solution and a large initial search space the necessary memory storage is small.

As a matter of fact, the described algorithm will usually not be very efficient, but there are numerous ways to improve it as will be shown later on. However, note that there is an easy way to improve the computation time of the basic algorithm; indeed, we may notice that each box in $\mathcal{L}$ is submitted to a processing that does not depend upon the other boxes. Hence it is possible to implement the algorithm in a distributed manner. A master computer will send to n slave computers the first n boxes in the list. These slave computers will individually perform a few iterations of the basic algorithm and will send back to the master the remaining boxes in its $\mathcal{L}$ list (if any) and the solutions it has found (if any). The master will manage a global list $\mathcal{L}$ and perform a few iterations of the basic algorithm if all the slaves are busy. We will discuss the efficiency of the distributed implementation in the Example sections.

3. Equations for the Forward Kinematics

Let A_i and B_i be the attachment points of the leg i on the base and on the platform, respectively. The coordinates of A_i in the reference frame will be denoted $x_{a_i}, y_{a_i}, z_{a_i}$ while the coordinates of B_i in the same frame are x_i, y_i, z_i. Without lack of generality we may choose $x_{a_1} = y_{a_1} = z_{a_1} = 0$ and $y_{a_2} = z_{a_2} = 0$. Note that for a given robot and given leg lengths it is always possible to change the numbering of the leg lengths and we will see that this has an influence on the computation time of our algorithm.

There are numerous ways to write the equations of the inverse kinematics (which constitute the system of equations to be solved for the FK problem) according to the parameters that are used to represent the pose of the platform. In this paper a pose of the platform will be defined either by the coordinates of the three points B_1, B_2, B_3 (assumed to be not collinear; such a triplet can always be found otherwise the robot is always singular) if the platform is planar, or by the coordinates of the four points B_1, B_2, B_3, B_4 in the general case. The chosen points will be denoted the reference points of the system, and the associated legs the reference legs. If m, $m \in [3, 4]$ points are used for defining the pose of the platform then for any j in $[m + 1, 6]$ we have

$$\mathbf{OB_j} = \sum_{k=1}^{k=m} \alpha_j^k \mathbf{OB_k}, \tag{3}$$

where α_j^k are known constants such that $\sum_{k=1}^{k=m} \alpha_j^k = 1$. A first set of equations is obtained by expressing the leg lengths for the m chosen reference legs

$$(x_j - x_{a_j})^2 + (y_j - y_{a_j})^2 + (z_j - z_{a_j})^2 = \rho_j^2, \quad j \in [1, m], \tag{4}$$

where ρ_j is the known leg length.

A second set of equations is obtained by writing the leg lengths for the legs $m + 1$ to 6, using the coordinates of the B_j points defined in eq. (3):

$$\begin{aligned} &\left(\sum_{i=1}^{i=m} \alpha_j^i x_i - x_{a_j}\right)^2 + \left(\sum_{i=1}^{i=m} \alpha_j^i y_i - y_{a_j}\right)^2 \\ &\quad + \left(\sum_{i=1}^{i=m} \alpha_j^i z_i - z_{a_j}\right)^2 = \rho_j^2, \quad j \in [m + 1, 6]. \end{aligned} \tag{5}$$

The third set of equations is obtained by writing that the distance between any couple of reference points $B_1, \ldots, B_m$ is a known quantity

$$\begin{aligned} &(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = d_{ij}^2, \\ &\quad i, j \in [1, m],\ i \neq j, \end{aligned} \tag{6}$$

where d_{ij} is the distance between the points B_i and B_j. It may be noted that eqs. (4), (5) and (6) are a set of distance equations which describe the distance between either points in the three-dimensional (3D) space or virtual points, i.e., points whose coordinates are a linear combination of reference points (here points $B_{m+1}, \ldots, B_6$ are the virtual points).

We end up with a system of $n = 3m$ equations in the $3m$ unknowns (x_i, y_i, z_i). It appears that for each equation in the system (4), (5) and (6) there is only one occurrence of each unknown. Consequently, according to Property A the interval evaluation of each equation gives the exact minimum and maximum values of the equations and this motivates the use of such representation of the pose of the platform.

It must be noted, however, that if $m = 4$ we have not a minimal parametrization of the system as only three points are needed to define the pose of the platform. Indeed for point B_k with k in [4,6] we have

$$\mathbf{B_1B_k} = \mu_1^k \mathbf{B_1B_2} + \mu_2^k \mathbf{B_1B_3} + \mu_3^k \mathbf{B_1B_2} \times \mathbf{B_1B_3}, \tag{7}$$

where the μ_i^k are known constants. Hence we are dealing with a system with twelve unknowns while the same problem may be stated in terms of only nine unknowns. Equations 7 may have been used for the FK problem in the general case to obtain the coordinates of B_4, B_5, B_6 as functions of the coordinates of B_1, B_2, B_3, thereby leading to a system of nine equations in nine unknowns. However, eq. (7) have the drawback in terms of interval analysis that there are multiple occurrences of the same variable in the equation. Hence, for the general case it is better to have more unknowns but no overestimation of the values of the equations. This is a general trend for the interval analysis based method in which it is often interesting to consider a simple expression even at the price of a larger number of unknowns.

4. Improving the Basic Solving Algorithm

We have presented in Section 2.2 a basic solving algorithm. The purpose of the following sections is to propose various methods that can be used to improve the efficiency of this algorithm.

4.1. Filtering the Boxes

A first possible way to improve the basic algorithm is to try to decrease the width of the current box "in place", a method which is called filtering in the constraints programming community. There are numerous filters that can be used, but we will only present the methods that may be of interest for the FK problem.

4.1.1 Constraint Propagation

Constraint propagation (Jaulin et al. 2001) is a classical method in the field of interval analysis. Without going into the details, its purpose is to use the constraints to filter the boxes, i.e., either to try to reduce their width or even to eliminate them.

A first filtering method is the 2B approach, a derivation of the k-B method introduced in Collavizza, Deloble, and Rueher (1999). Let us consider, for example, eq. (4):

$$(x_j - x_{a_j})^2 + (y_j - y_{a_j})^2 + (z_j - z_{a_j})^2 = \rho_j^2. \qquad (8)$$

We may rewrite this equation as

$$(x_j - x_{a_j})^2 = \rho_j^2 - (y_j - y_{a_j})^2 - (z_j - z_{a_j})^2.$$

Let U_1 and U_2 be the interval evaluations of the left and right terms of this equation. The lower bound of the interval evaluation U_1 is obtained as

$$\underline{U_1} = \begin{cases} 0 & \text{if } \underline{x_j} - x_{a_j} \le 0 \text{ and } \overline{x_j} - x_{a_j} \ge 0 \\ (\overline{x_j} - x_{a_j})^2 & \text{if } \overline{x_j} - x_{a_j} \le 0 \\ (\underline{x_j} - x_{a_j})^2 & \text{if } \underline{x_j} - x_{a_j} \ge 0 \end{cases}$$

while the upper bound is

$$\overline{U_1} = \text{Max}((\underline{x_j} - x_{a_j})^2, (\overline{x_j} - x_{a_j})^2).$$

Clearly, if eq. (8) has a solution for the current box, then the left term value at the solution will be included in $U_3 = U_1 \cap U_2$. Three cases may occur:

1. $U_1 \cap U_2 = \emptyset$;
2. $U_1 \subseteq U_2$;
3. $U_1 \cap U_2 \subset U_1$.

If $U_1 \cap U_2 = \emptyset$, then the equation has no solution and the current box can be rejected. If $U_1 \subseteq U_2$, nothing is done. If $U_1 \cap U_2 \subset U_1$, we may have either $\overline{U_3} < \overline{U_1}$ and/or $\underline{U_3} > \underline{U_1}$. We assume that $\overline{U_3} < \overline{U_1}$. This implies that

$$(x_j - x_{a_j})^2 \le \overline{U_3}$$

or

$$-\sqrt{\overline{U_3}} + x_{a_j} \le x_j \le \sqrt{\overline{U_3}} + x_{a_j}.$$

Hence, the lower or upper bounds for x_j may be updated. We assume now that $\underline{U_3} > \underline{U_1}$ and $\underline{U_3} > 0$. This implies

$$x_j \le x_{a_j} - \sqrt{\underline{U_3}} \text{ or } x_j \ge x_{a_j} + \sqrt{\underline{U_3}},$$

which may allow the range for x_j to be narrowed.

As a simple example, consider the equation

$$x_1^2 + y_1^2 + z_1^2 = 100$$

with x_1 in [0,10] and y_1, z_1 in the range [4,6]. Applying the 2B method on x_1 we obtain

$$\begin{aligned} U_1 &= x_1^2 = [0, 100] \\ U_2 &= 100 - y_1^2 - z_1^2 = 100 - [16, 36] - [16, 36] \\ &= [28, 68]. \end{aligned}$$

The intersection of U_1, U_2 is [28,68] and hence x_1^2 must lie in this range. Hence, we obtain that x_1 must lie in the range $[\sqrt{28}, \sqrt{68}]$, i.e., approximately [4.24,8.24]. Hence the new width of the range for x_1 is 4 while the width of its initial range was 10; a simple arithmetic operation has allowed us to reduce the search space for x_1 by 60%.

This process may be repeated for each unknown in the equation and a similar process may be used for equations of type (5) and (6). Note also that, as soon as the range for a variable has been changed, it may be interesting to restart the process from the beginning as new variables may be updated. There is, however, a limit to this process as the convergence of this method is asymptotically slow. In our method, the process is repeated only if the change of the width of the range for one unknown is greater than a fixed threshold and the 2B method is

used only on the equations that include an updated unknown. If we are using four reference points, the 2B method based on eq. (7) will also be used to filter boxes.

A second filtering method is the 3B method. In this approach the range $[x_j, \overline{x_j}]$ for one variable x_j in a given box B is replaced by $[\underline{x_j}, \underline{x_j}+\epsilon]$, where ϵ is an arbitrary small number, while the ranges for the other unknowns are unchanged. We then test whether, for this new set of ranges, the system may have some solution either by using the 2B method and/or by evaluating the equations. If the answer is negative, the range for x_j in the box B may be changed to $[\underline{x_j}+\epsilon, \overline{x_j}]$. The process is then repeated on the new range but the change in the range will be doubled, i.e., we will test the range $[\underline{x_j}, \underline{x_j}+2\epsilon]$. The process is then repeated until the no-solution test is no longer satisfied. At this point we are testing the range $[\underline{x_j}, \underline{x_j}+k\epsilon]$, where k is a positive even integer. If $k > 1$ we may either repeat the process with a change ϵ or stop the procedure. Note, however, that as soon as the range for one variable has been modified it may be interesting to use the 2B method to eventually update the range for all the unknowns.

Up to now, we have tried to decrease the width of the range for x_j by increasing its lower bound. Clearly the process may be repeated on the "right" side by trying to decrease the upper bound of the range for x_j (i.e., we will test the range $\overline{x_j}-\epsilon, \overline{x_j}]$). In the same manner the process will be repeated in turn for each unknown.

4.1.2. Global Filtering Methods

A drawback of the previously mentioned methods is that they are local, i.e., they are working on each equation of the system but are not considering the whole set of equations.

Global filtering methods that consider at least two equations may be designed and we will present now two such methods. The first global filtering method is inspired from Yamamura, Kawata, and Tokue (1998). Let us consider again the equation

$$(x_j - x_{a_j})^2 + (y_j - y_{a_j})^2 + (z_j - z_{a_j})^2 = \rho_j^2. \tag{9}$$

We will use the change of variable

$$a_j = x_j - \underline{x_j} \qquad b_j = y_j - \underline{y_j} \;\; c_j = z_j - \underline{z_j}.$$

Hence the ranges for these new variables are

$$[0, \overline{x_j} - \underline{x_j}], \qquad [0, \overline{y_j} - \underline{y_j}], \qquad [0, \overline{z_j} - \underline{z_j}].$$

Equation (9) may be written as

$$(a_j - x_{a_j} + \underline{x_j})^2 + (b_j - y_{a_j} + \underline{y_j})^2 + (c_j - z_{a_j} + \underline{z_j})^2 = \rho_j^2. \tag{10}$$

Expanding this equation we obtain

$$a_j^2 + b_j^2 + c_j^2 + U_1 a_j + U_2 b_j + U_3 c_j + U_4 = 0 \tag{11}$$

where, for a given box, the U terms are constant. Let define α_j as $a_j^2 + b_j^2 + c_j^2$. Using interval arithmetics it is possible to determine an interval for α_j=$[\underline{\alpha_j}, \overline{\alpha_j}]$. Introducing α_j in eq. (11) it becomes a linear equation in terms of the unknowns α_j, a_j, b_j, c_j while the unknown α_j is submitted to the linear inequalities

$$\underline{\alpha_j} \le \alpha_j \le \overline{\alpha_j}. \tag{12}$$

This process is repeated for each of the n equations (4), (5) and (6), and we end up with a linear system of n equations in terms of the n unknowns a_j, b_j, c_j, $j \in [1, 3]$ and n additional unknowns α_k which represent the non-linear part of each equation. Furthermore, these n additional unknowns are submitted to the $2n$ linear inequalities (12). A direct consequence of this linearization is that we may think to use the first step of the simplex algorithm to determine if this linear system admits a feasible region. If this is not the case, the non-linear equations (4), (5) and (6) have no solution and the box is rejected. However, if a feasible region is detected, we can still use the second step of the simplex method that allows us to minimize or maximize a linear cost function. Here we will in turn determine the minimum and maximum of each variable a_j, b_j, c_j. If the minimum (maximum) is larger (lower) than the current bound for the variable, then this bound is updated as α_j is updated, leading to a new linear system. This process is repeated until the change obtained for the bounds is lower than a predefined threshold. Note that we use an interval variant of the simplex method in order to ensure that round-off errors do not have an impact on the result.

The second global filtering method is the classical interval Newton method. Let a system of n equations in n unknowns

$$F = \{F_i(x_1, \ldots, x_n) = 0, i \in [1, n]\}$$

and consider the following iterative scheme

$$X_{k+1} = (M(X_k) - A\ F(M(X_k))) \cap X_k = U_k \cap X_k, \tag{13}$$

where A is an interval matrix that contains all the inverse of the Jacobian matrix of the system F (we will not explain in detail how to compute such a matrix). It is possible to show the following properties of this scheme:

- if $U_k \cap X_k = \emptyset$, then the system F has no solution in X_k;
- if $U_k \cap X_k \subset X_k$, then solutions of F in X_k are also in X_{k+1}.

We use in fact the Hansen–Sengupta version of the interval Newton method, which uses a pre-conditioning of the Jacobian matrix by its inverse at the middle point of the box, to improve the interval inputs (see Ratscheck and Rokne 1995).

4.2. Unicity Operators

The purpose of these operators is to determine eventually a box, called a unicity box, which contains a unique solution of the system, and which furthermore can be numerically computed in a certified manner. Two such methods are used in our algorithm.

4.2.1. Kantorovitch Theorem

Let a system of n equations in n unknowns

$$F = \{F_i(x_1, \ldots, x_n) = 0, i \in [1, n]\}.$$

Let $\mathbf{x_0}$ be a point and $U = \{\mathbf{x}/||\mathbf{x} - \mathbf{x_0}|| \leq 2B_0\}$, the norm being $||A|| = \text{Max}_i \sum_j |a_{ij}|$. We assume that $\mathbf{x_0}$ is such that

1. the Jacobian matrix of the system has an inverse Γ_0 at $\mathbf{x_0}$ such that $||\Gamma_0|| \leq A_0$;
2. $||\Gamma_0 F(\mathbf{x_0})|| \leq 2B_0$;
3. $\sum_{k=1}^{n} |\frac{\partial^2 F_i(\mathbf{x})}{\partial x_j \partial x_k}| \leq C$ for $i, j = 1, \ldots, n$ and $\mathbf{x} \in U$;
4. the constants A_0, B_0, C satisfy $2nA_0B_0C \leq 1$.

Then there is a unique solution of $F = 0$ in U and the Newton method used with $\mathbf{x_0}$ as an estimate of the solution will converge toward this solution (Tapia 1971).

Conversely let us compute B_1 as $1/(2nA_0C)$ and assume that $B_1 \leq B_0$. Then there is a unique solution in the box $[x_0 - 2B_1, x_0 + 2B_1]$. This is the general version of the Kantorovitch theorem but the system we are dealing with is specific. Indeed, the Hessian matrix is constant as we are dealing with quadratic equations; a direct consequence is that the C term can be pre-computed. A further consequence is that we have been able to show that the factor n in B_1 may be substituted by the dimension of the space in which are written the distance equations (i.e., three for the current problem), thereby enlarging the box in which a unique solution is found. In our algorithm, the Kantorovitch theorem is used for each box having a width lower than a given threshold, with x_0 being the center of the box. This may allow us to find a ball centered at x_0 that contains a unique solution.

4.2.2. Inflation Operator

The second unicity operator is based on the use of the Newton scheme. For each box we run a few iterations of the Newton method with as an initial guess the center of the box. If after a few iterations the scheme has converged to a point $\mathbf{X}$, for which the absolute values of all the equations are lower than a small threshold κ, we will first verify that there is a ball $\mathcal{B}_K$ centered at $\mathbf{X}$ that contains a solution of the FK problem by using the Kantorovitch theorem applied at $\mathbf{X}$. However, the diameter of this box will usually be very small as its maximum is the norm of $||\Gamma_0 F(\mathbf{X})||$ with all elements of $F(\mathbf{X})$ being lower than κ in absolute value. We will now explain how to calculate a box centered at $\mathbf{X}$, that contains only one solution of the system but which will have, in general, a much larger diameter than $\mathcal{B}_K$.

Let us assume that X_0 is a solution of a system F and that X_1 is another solution close to X_0, hence $F(X_0) = F(X_1) = 0$. Using the mean value theorem we obtain $F(X_1) = F(X_0) + J(X)(X_1 - X_0)$ where J is the Jacobian matrix of the system and X lies in the box $[X_0, X_1]$. Hence we obtain that $J(X)(X_1 - X_0) = 0$ which admits a solution only if $J(X)$ is singular. The principle is now to obtain a box centered at X_0 such that $J(X)$ is regular for any point in the box. To obtain such a box we use the H-matrix theory of Neumaier (1990, 2001). We will explain first an implementation of this scheme that will work whatever is the system of n equations in n unknowns we are considering.

Let us consider a box $\mathcal{B}$ $[X_0 - \epsilon, X_0 + \epsilon]$ centered at X_0 and compute the interval Jacobian matrix for this box that we multiply by the inverse of the Jacobian matrix computed at X_0 to obtain an interval matrix $S = ((S_{ij}))$. Consider the diagonal element S_{ii} having the lowest mignitude s_{ii} and let us define m_j as the sum of the magnitude of the intervals in the jth row of S, excluding the diagonal element of the line, while M_S will be the maximum of the m_j over the rows of S. If $s_{ii} > M_S$, then J is regular over the whole $\mathcal{B}$.

If the regularity test is satisfied for $\mathcal{B}$, then we expand it to $[X_0 - 2\epsilon, X_0 + 2\epsilon]$ until the regularity test is not satisfied for the box $[X_0 - 2^n\epsilon, X_0 + 2^n\epsilon]$. If the regularity test is not satisfied for $n = 1$, then we will use the box $\mathcal{B}_K$ as a unicity box.

We may, however, specialize this scheme in the particular case of distance equations. Indeed, in that case each component of the Jacobian matrix is linear in terms of the unknowns. Let $\{x_i^0\}$ be the elements of X_0, let J_0^{-1} be the inverse of the Jacobian matrix computed at X_0 and let X_1 be defined as $\{x_i^0 + \kappa\}$, where κ is the interval $[-\epsilon, \epsilon]$. Each component J_{ij} of the Jacobian at X_1 can be calculated as $\alpha_{ij} + \beta_{ij}\kappa$, where α_{ij}, β_{ij} are constants which depend only upon X_0. If we multiply J by J_0^{-1} we obtain a matrix $U = J_0^{-1}J = I_n + A$, where I_n is the identity matrix of dimension n and A is a matrix such that $A_{ij} = \zeta_{ij}\kappa$ where ζ_{ij} can be calculated as a function of the β coefficients and of the components of J_0^{-1}. For a given line i of the matrix U the diagonal element has a mignitude $1 - |\zeta_{ii}|\epsilon$ while the sum of the magnitude of the non-diagonal element is $\epsilon \sum_{j=1}^{j=n} |\zeta_{ij}|$, $j \neq i$. The matrix U will be guaranteed to be regular if for all i

$$\epsilon \sum_{j=1}^{j=n} |\zeta_{ij}| \ (i \in [1, n], \ j \neq i) \ \leq 1 - |\zeta_{ii}|\epsilon$$

which leads to

$$\epsilon \leq \frac{1}{|\zeta_{ii}| + \text{Max}(\sum_{j=1}^{j=n} |\zeta_{kj}|), k \in [1, n], j \neq k}. \qquad (14)$$

Hence, the minimal value ϵ_m of the right term of this inequality over the lines of U allows us to define a box $[X_0-\epsilon_m, X_0+\epsilon_m]$ which contains a unique solution of the system. In general, this box will be larger than the box computed with the Kantorovitch theorem. Note that the round-off errors involved in the computation of J_0^{-1} may lead to a wrong estimate of ϵ_m. However, we can perform the regularity test with this value and decrease it by a small amount if this test fails.

Ultimately the Kantorovitch scheme may fail if two (or more) solutions are very close and the round-off error does not allow us to separate them. In that case the algorithm will still stop as we give a minimum threshold on the width of the box. If the width of a box is lower than this threshold and the interval evaluations of the equations for this box all include 0, then this box is considered a solution. However, if a solution is returned as a box and not as a point interval it indicates that a further analysis has to be performed around this interval solution; either we are close to a singularity (this can be detected safely using the scheme proposed in Merlet and Daney 2001) or we have a cluster of solutions that can only be calculated by using an extended arithmetics such as MPFI.

4.3. Filtering with the Unicity Box

Let us assume that a unicity box $\mathcal{B}^u$ has been found by one of the methods proposed in the previous section. We will propagate this knowledge in the boxes still to be processed to avoid finding again the same solution. Indeed, if there is an intersection between a box B_k and $\mathcal{B}^u$, then new solutions can only be found in $B_k - (B_k \cap \mathcal{B}^u)$. For a box B_k in the list two cases may occur:

1. $B_k \subset \mathcal{B}^u$ – the box B_k is eliminated from the list;
2. $\mathcal{B}^u \subset B_k$ or $\mathcal{B}^u \not\subset B_k$ but $\mathcal{B}^u \cap B_k \neq \emptyset - B_k - (B_k \cap \mathcal{B}^u)$ is calculated. This calculation may lead to at most $2m$ new boxes but this should be avoided. So we first filter the new boxes using the algorithms proposed in Section 4.1 and we impose a limit on the number of new boxes (typically no more than four new boxes may be created).

4.4. The Improved Algorithm

The basic solving algorithm is hence improved by using the various methods described in the previous sections. The added steps are presented in italic font, followed by the section number that describes the step. By convention a step that allows us to eliminate a box will return -1, while a step that determines a solution will return 1.

1. If $i > n$, then return the solution list.
2. *If solutions have been found filter B_i with the unicity box filter* (4.3).
3. If *local filter* = –1, then $i = i + 1$ and go to 1 (4.1).
4. If *global filter* = –1, then $i = i + 1$ and go to 1 (4.1.2).
5. *If unicity method* = 1 (4.2), then
   (a) test if the solution has not already been found;
   (b) if not add the solution to the solution list;
   (c) use the unicity box filter on B_i (4.3).
6. If at least one $F_j(B_i)$ exists such that $0 \notin F_j(B_i)$, then $i = i + 1$ and go to 1.
7. If $0 \in F_j(B_i)\ \forall\ j \in [1, n]$ and $w(F_j(B_i)) \leq \epsilon$, then store B_i in the solution list, $i = i + 1$ and go to 1.
8. Select the unknown k which has the largest width in B_i. Bisect its interval in the middle point and create two new boxes from B_i as $X_1, \ldots, X_{k-1}, [\underline{X_k}, (\underline{X_k} + \overline{X_k})/2], \ldots, X_n]$ and $X_1, \ldots, X_{k-1}, [(\underline{X_k}+\overline{X_k})/2, \overline{X_k}], \ldots, X_n]$. Store these boxes as B_{n+1}, B_{n+2}, $n = n + 2$ and go to 1.

A further refinement may be added. Indeed, it is interesting to determine as soon as possible all the solutions of the system as the filtering with the unicity box described in the previous section will decrease the search space. A pure depth first storage mode is not very efficient in this view. As the filtering described in Section 4.2 may allow us to determine the solution of the system even for boxes with a large width, we will change the ordering of the boxes in the list $\mathcal{L}$. After a fixed number of iterations of the algorithm, the current box B_i will be substituted by the box having the largest width among the boxes that have to be processed.

5. Determining the Search Space

Interval analysis is, in general, used for problems where ranges for the solution may be specified. Furthermore, the efficiency of these methods is usually very sensitive to the size of the search space.

Fortunately the FK problem belongs to the category of problems for which bounds for the solutions are easily found. We will describe here simple methods that allow us to determine an initial search space.

5.1. Bounds from a Single Leg

Obtaining bounds for the n unknowns is straightforward when considering the constraint on one leg. Let d_{ij} be the known distance between B_i and B_j and define for i, j in $[1, m]$:

$$\begin{aligned} p_{ij} &= x_{a_j} + \rho_j + d_{ij} \\ q_{ij} &= y_{a_j} + \rho_j + d_{ij} \\ r_{ij} &= z_{a_j} + \rho_j + d_{ij}. \end{aligned}$$

Clearly, for a given j the x, y, z coordinates of B_i cannot exceed p_{ij}, q_{ij}, r_{ij} and be lower than $-p_{ij}, -q_{ij}, -r_{ij}$. Hence, an initial search space for the coordinates of B_i is the box defined as

$$[\text{Max}(-p_{ij}), \text{Min}(p_{ij})], [\text{Max}(-q_{ij}), \text{Min}(q_{ij})],$$

$$[\text{Max}(-r_{ij}), \text{Min}(r_{ij})],$$

where $i, j \in [1, m]$.

5.2. Improved Bounds for a Pair of Legs

Let us consider two legs with extreme points B_i, B_j and denote D_{ij} as the distance between A_i and A_j. Point B_i is constrained to lie on a sphere S_1 centered at A_i with radius ρ_i. At the same time, this point is constrained to lie inside a sphere S_2 centered at A_j with radius $\rho_j + d_{ij}$ and, if $\rho_j \geq d_{ij}$, to lie outside a sphere S_3 centered at A_j with radius $\rho_j - d_{ij}$. Hence, four different cases may occur for B_i (Figure 2):

1. If $\rho_j + d_{ij} > \rho_i + D_{ij}$ and $\rho_j - d_i j \leq D_{ij}$, point B_i will lie on the sphere S_1.
2. If $\rho_j + d_{ij} > \rho_i + D_{ij}$ and $\rho_j - d_i j \geq D_{ij}$, point B_i will lie on the part of the sphere S_1 bordered by the intersection between S_1, S_3 which is the farthest from A_j.
3. If $\rho_j + d_{ij} < \rho_i + D_{ij}$ and $\rho_j - d_i j \leq D_{ij}$, point B_i will lie on the part of the sphere S_1 bordered by the intersection between S_1, S_2 and which is the closest to A_j.
4. If $\rho_j + d_{ij} < \rho_i + D_{ij}$ and $\rho_j - d_i j \geq D_{ij}$, point B_i will lie between the part of the sphere S_1 the closest to A_j delimited by the intersection between S_1, S_2 and the part of the sphere S_1 the closest to A_j delimited by the intersection between S_1, S_3.

In the first case, no further information on the bound for the coordinate of B_i compared to the bound found in the previous section will be obtained. For the three other cases, we assume first that the direction A_i, A_j is the direction of the x-axis of the reference frame. For the second case, we assume that C_3 is the circle projection of S_3 in the x, z plane and let N_1, N_2 be the intersection points between C_1, C_3, both having the same x coordinate x_N. Then, x_N is an upper bound for the x coordinate of B_i. For the third case, let C_1, C_2 be the circle projection of the sphere S_1, S_2 in the x, z plane and let M_1, M_2 be the two intersection points of C_1, C_2 (which have the same x coordinate x_M). Clearly, x_M is a lower bound for the x coordinate of B_i which is a better lower bound than $-p_{ij}$ found in the previous section. Furthermore, if $\rho_j + d_{ij} < D_{ij}$ then z_M; the z coordinate of M is an upper bound for the z coordinate of B_i while $-z_M$ is a lower bound. As there is a circular symmetry in the problem, $-z_M$ and z_M are also lower and upper bounds for the y coordinate of B_i. As in the second case, x_N is an upper bound for the x coordinate of B_i.

Using this procedure a new bounding box $\mathcal{B}$ is obtained for B_i. Now, if A_i, A_j are not on the x-axis of the reference frame there is a rotation matrix $\mathcal{R}$ that allows us to convert a vector in our bounding box frame to a vector in the reference frame. Applying this rotation matrix on $\mathcal{B}$ will allow us to obtain a bounding box in the reference frame and update accordingly the bounds for the n unknowns. This process may be repeated for each pair of legs.

5.3. Numbering the Legs

It must be noted that the numbering of the legs may be changed arbitrarily. A change in this numbering has first an effect on the size of the search space. For example, if we are using the two previous algorithms it is interesting to reorder the numbering of the legs so that the selected leg 1 will have the lowest leg length while the legs 2 and 3 will be those presenting the lowest absolute value for $\rho_i + d_{i1}$ among the five remaining legs. This allows us to reduce the size of the search space with a minimal amount of computation time. However, it must also be noted that the coefficients α_j^i appearing in the FK equations play an important role as they increase or decrease the range for the coordinates of the virtual points. We will see in the section devoted to the results that the choice of the numbering has a drastic effect on the computation time.

6. Extensions to the Improved Algorithm

The algorithm presented in the previous sections has two good features:

- any additional constraints that may limit the number of realistic solutions to the FK problem may be taken into account with a direct impact on the computation time;
- uncertainties on the robot can be taken into account.

We will exemplify these features in the next two sections.

6.1. Range of Motion for the Passive Joints

An advantage of the proposed algorithm is that it can easily be modified to take into account technological constraints that will limit the number of solutions that can effectively be reached by the robot. For example, the passive joints at A_i, B_i may have a limited range of motion. Each joint is such that the angle θ between the leg connected to the joint and a known direction (defined by a unit vector $\mathbf{u_i}$, the reference axis of the joint) must be less than a given value θ_{max}.

For the joint at A_i this may be written as

$$\frac{\mathbf{A_i B_i}}{\rho_i}.\mathbf{u_i} \geq \cos\theta_{max}. \tag{15}$$

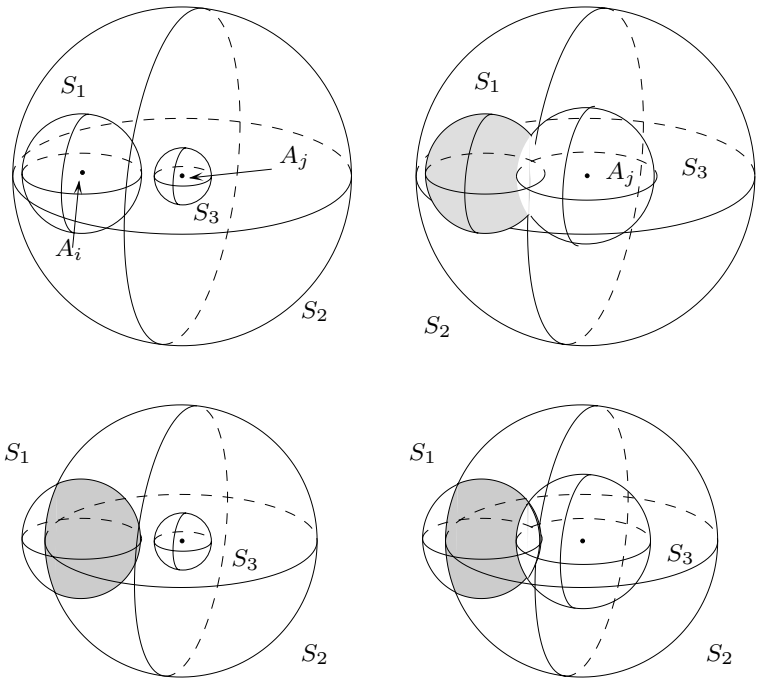

Fig. 2. The four different cases for the possible location of B_i related to the location of A_j, the leg length ρ_j and the distance between B_i, B_j. The allowed zone for B_i on the sphere centered at A_i with radius ρ_i is presented in gray.

In this equation the components of the vector $\mathbf{A_iB_i}$ can be expressed as a linear function of our unknowns while ρ_i, $\mathbf{u_i}$ are known quantities. Hence, we are able to evaluate for each box the left term U_i of the previous inequalities. An additional filter will then be used in our algorithm:

- For the box evaluate U_i and if $\overline{U_i} < \cos\theta_{max}$, then eliminate the box as it cannot contain a solution that will respect the joint constraints.

- The left term of the inequality can be expressed as $\sum_{k=1}^{k=r} \mu_k X_k$ where X_k are unknowns of the FK problem. The inequality can be written as $\mu_1 X_1 \geq \cos\theta_{max} - \sum_{k=2}^{k=r} \mu_k X_k$. Let W_1 be the interval evaluation of the right term of this inequality and assume that μ_1 is positive. We find that $\underline{X_1}$ must be greater or equal to $\underline{W_1}/\overline{mu_1}$, which may allow us to update the range for $\overline{X_1}$. A similar process may be used to improve the ranges of the other variables.

- Eliminate the solutions found in Section 4.2 that do not satisfy the joint constraints.

For the joint at B_i the reference axis $\mathbf{u_i}$ is no longer fixed in the reference frame. Hence its components have to be established as functions of the unknowns. There are constants β_i^k such that

$$\mathbf{u_i} = \beta_i^1 \mathbf{B_1B_2} + \beta_i^2 \mathbf{B_1B_3} + \beta_i^3 (\mathbf{B_1B_2} \times \mathbf{B_1B_3}),$$

as we have assumed that B_1, B_2, B_3 were not collinear. A similar filter for the limited motion of the joints at B_i can thus be designed using eq. (15).

Note that taking into account the joint motion limitation within the calculation allows us to reduce the computation time and is better than computing all the solutions and then sorting them.

6.2. Modeling Uncertainties

Assume now that some parameters in the model of the robot are not perfectly known; for example, the location of the joints at A_i, B_i, the leg lengths, ... may be known only up to a given accuracy, usually relatively small. Up to now, we have assumed that these N parameters Q_j have a fixed value while in fact they lie in known ranges I_Q^j. Note that very often the real values of a parameter will indeed cover the full range I_Q^j; for example, if we have clearance in the joints at A_i, B_i the location of the A_i, B_i will depend upon the static equilibrium of the robot.

Under this assumption the FK problem does not have a finite number of solutions but is a manifold of solutions as the coefficients of the FK equations are now intervals. However,

we will still be able to determine an approximation of this manifold, i.e., find all the poses that can be reached by the robot.

We will introduce as additional unknowns the N variables Q_j (for example, Δx_{a_i}, Δy_{a_i}, Δz_{a_i} for the location of the A_i points). The FK equation becomes a system of n equations in the $N + n$ variables, but the algorithm can still be used to solve this system provided that:

- the unicity filter (Section 4.2) is no longer used as usually it will not be possible to extract a square system in the n equations, but it may still be used if we consider that the FK equations have interval coefficients (in that case the unicity filter will provide an outward hull of the manifold);
- a solution is supposed to be found as soon as the width of a box is lower than ϵ while the interval evaluation of the equations still contain 0 – these boxes are written in a file and their total volume is V_s;
- boxes with a width lower than α, a value provided by the user, are eliminated from the list of boxes – they are called the neglected boxes and their total volume is V_n.

Using this method we will be able to calculate an approximation of the set of all solutions of the FK problem whatever the values of the physical parameters of the real robot. This approximation is an inward hull of the manifold.

The total volume of the region that can be reached by the robot will not exceed $V_s + V_n$ and the ratio V_s/V_n is a good index to determine the quality of the approximation. Note also that this quality index can be improved incrementally. Indeed, we may store the neglected boxes obtained for a given value α_1 of α in a file (which has led to a solution volume V_s^1) and then process only the boxes in this file if a value $\alpha_2 < \alpha_1$ of α is selected. Indeed, there is no need to process again the whole search domain as we have already determined the solution volume V_s^1. For the j run of the algorithm with value $\alpha_j < \alpha_{j-1} < \ldots < \alpha_1$ for α, the solution volume will be $V_s^1 + V_s^2 + \ldots + V_s^j$ while the neglected volume V_n^j will be such that $V_n^j \leq V_n^{j-1} \leq \ldots \leq V_n^1$.

7. Implementation

The tests have been performed using the software ALIAS [4] which is a C++ library, available for SUN/Unix and PC/Linux, which implement most interval analysis methods described in the previous sections. Basic interval arithmetics operations are performed with the BIAS/Profil library (with a precision of double). An innovation of this package is that it is partially interfaced with Maple:

- the equations which are to be solved are defined in a Maple session;
- by using a specific Maple library the C++ code based on ALIAS which is necessary to solve the system is automatically produced and then is executed. Furthermore, the C++ code that is equation-dependent (such as the code for the 2B filtering method that is used when dealing with eq. (7)) is also produced;
- the result of the solving algorithm is made available in the Maple session;
- using the multi-precision ability of Maple, the accuracy of the obtained solutions (related to the accuracy of the C++ code) can be improved (solutions may be determined up to an arbitrary number of digits).

Note that it is possible to use this Maple interface to produce a generic C++ program that may be used for a given robot but for any leg lengths. Hence, the computation time given in the following section will be the execution time of the generic program.

ALIAS also allows us to use the distributed implementation of the solving algorithm within the Maple session. Basically only the names of slave computers have to be given and a distributed implementation will be created using the message passing mechanisms of the parallel virtual machine (PVM; Geist 1994).

8. Example 1

Here, we will use the FK problem presented in Lee and Shim (2001b). The coordinates of the A_i points are

$$A_1\ [0,0,0]\ \ A_2\ [62,0,0]\ \ A_3\ [7,13,0]$$
$$A_4\ [42,38,0]\ \ A_5\ [32,39,0]\ \ A_6\ [62,11,0]$$

while the coordinates of the B_i are

$$B_1\ [0,0,0]\ \ B_2\ [14,0,0]\ \ B_3\ [16,42,0]$$
$$B_4\ [46,27,0]\ \ B_5\ [23,45,0]\ \ B_6\ [47,13,0].$$

The length of the legs are

$$\rho_1 = 99.44345126\ \ \rho_2 = 122.3824766$$
$$\rho_3 = 119.2390086$$
$$\rho_4 = 153.9499536\ \ \rho_5 = 136.2700605$$
$$\rho_6 = 156.0149565.$$

Here, the platform is planar and hence we have nine unknowns. Note that due to the symmetry with respect to the base we will always obtain an even number of solutions (for each solution with B_1 over the base we will obtain another solution which is the mirror of the first one with B_1 under the base).

4. http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-C++/ALIAS-C++.html and http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-Maple/ALIAS-Maple.html.

8.1. Numbering the Legs and Computation Time

We may select arbitrarily the numbering of the legs by choosing three legs that will be numbered from 1 to 3 among the six possible legs; hence there is a total of 20 possible numbering. Two factors may be modified when changing the numbering: the search space and the values of the α_j^i coefficients. As the amount of time for computing the search space and the α_j^i coefficients is very low, all combinations can be considered. The following elements can be calculated: mean value M_s of the search space diameter ranges, minimal diameter I_{min} for the range of the search space, mean value M_α of the absolute value of the nine α_j^i coefficients, and mean value M_6 of the diameter of the ranges for the six attachment points B_i. For the current example, we obtain data provided in Table 1.

The values of M_6 allow us to distinguish three main groups: one having a value M_6 less than 200, one having a value between 200 and 300 and the remaining one having a value larger than 400. It seems reasonable to keep only as possible numbering the elements of the first group as a large M_6 will impose a large number of bisection to satisfy eq. (5). A second criterion is that M_s should be minimal as the bisection will be applied on the intervals that are used to compute this mean value. Using both criteria, the most promising combinations are (2,3,6), (2,3,5), (1,3,5), (1,3,6), (1,4,6), and (1,4,5), in this order.

The tests have been performed using the standard Maple interface of the ALIAS library on a single PC laptop EVO 410 C, 2 GHz. The problem admits a total of four solutions and hence a total of two solutions in our search space. The solving times for finding all four solutions of the FK problem on the PC laptop for the six selected combinations are, respectively, 15, 17, 17, 17, 20, and 23 s.

Among all the combinations, the worst computation time is 420 s for the combination (1,5,6) which has the fourth largest M_s and the third largest M_6. If we have selected the combination having the lowest M_s (1,2,3) we obtain a computation time of 238 s (but this combination has the second largest M_6). If we rank the combination according to their values for M_s and M_6 and average the two ranks we again find that (2,3,6) is the best while (1,2,3) is among the worst. If we extend that to the nine best combinations, the worst computation time is 179 s for (2,3,4).

We have also tested the distributed implementation of our algorithm which is also available directly within the Maple interface. To average the performances, we have tested the combination (1,2,6) which has the second best M_s but the twelfth M_6 and for which the computation time on the laptop is 50 s.

The parallel implementation has been tested on a heterogeneous cluster of 16 low-level SUN and PCs that are shared by our team; the computation times vary between 5 and 15 s according to the load of the slave computers. Then, the same program has been tested on a cluster of 16 high-performance 2 Ghz PCs with a computation time of about 3–4 s.

Note that the distributed implementation allows a gain in computation time which may be larger than the number of machines despite the overhead time due to the message passing scheme. Indeed, in that case solutions may be found earlier in the process and the filtering process described in Section 4.3 allows us to eliminate the solution parts within the box that contain solutions, leaving only boxes that contain no solution and are quickly dismissed as such.

9. Example 2

In this section we will study the example provided in Dietmaier (1998) which has 40 solutions. The coordinates of the A_i points are

$$A_1\ [0, 0, 0] \quad A_2\ [1.107915, 0, 0]$$
$$A_3\ [0.549094, 0.756063, 0]$$
$$A_4\ [0.735077, -0.223935, 0.525991]$$
$$A_5\ [0.514188, -0.526063, -0.368418]$$
$$A_6\ [0.590473, 0.094733, -0.205018]$$

while the coordinates of the B_i are

$$B_1\ [0, 0, 0] \quad B_2\ [0.542805, 0, 0]$$
$$B_3\ [0.956919, -0.528915, 0]$$
$$B_4\ [0.665885, -0.353482, 1.402538]$$
$$B_5\ [0.478359, 1.158742, 0.107672]$$
$$B_6\ [-0.137087, -0.235121, 0.353913].$$

The leg lengths are

$$\rho_1 = 1 \quad \rho_2 = 0.645275 \quad \rho_3 = 1.086284$$
$$\rho_4 = 1.503439 \quad \rho_5 = 1.281933 \quad \rho_6 = 0.771071.$$

As the platform is not planar we use the formulation of the problem with twelve unknowns. Note that the solving parameters are exactly the same as in the previous example although the scale of this robot is quite different.

9.1. Numbering the Legs and Computation Time

As in the previous case the numbering of the legs has a large influence on the size of the search space and on the computation time. Here, the mean value of the ranges of the search space is more significant as the number of unknowns is larger than in the previous case.

An analysis similar to that performed in the previous section leads us to eliminate the combinations (2,3,5,6), (1,2,4,6), and (1,2,3,5) that have by far the largest M_6 (see Table 2).

If we select the six combinations that have the best M_s, i.e., (1,2,3,6), (1,2,5,6), (1,2,3,4), (1,3,5,6), (2,3,4,6), (1,3,4,6), we obtain respectively the following computation times: 22, 23,

Table 1. Evaluation of the Search Space Size According to the Numbering of the Legs

Combination	M_s	M_α	M_6	I_{min}
1,2,3	125.92	5.526	1064.08	44.9
1,2,4	146.32	2.346	554.717	44.9
1,2,5	138.77	1.324	331.64	44.9
1,2,6	137.05	1.443	352.63	44.9
1,3,4	148.57	1.37	392.4	89.16
1,3,5	150.92	0.4229	172.538	89.16
1,3,6	150.82	0.444	177.229	89.16
1,4,5	150.06	0.5319	194.78	67.71
1,4,6	150.15	0.517	191.79	73.86
1,5,6	151.074	3.85	970.05	69.97
2,3,4	139.89	1.67	435.78	44.9
2,3,5	139.99	0.531	180.44	44.9
2,3,6	139.16	0.543	181.83	44.9
2,4,5	196.75	0.591	273.57	108.87
2,4,6	153.96	0.55	204.04	108.87
2,5,6	156.59	2.76	721	87.219
3,4,5	142.07	2.87	683.59	91.58
3,4,6	143.9	2.15	536.42	96.2
3,5,6	149.07	2.908	719.349	54.08
4,5,6	200.87	3.974	1269.98	150.69

The combination indicates which legs are the reference legs 1, 2 and 3 among the six possible legs.

Table 2. Evaluation of the Search Space Size According to the Numbering of the Legs

Combination	M_s	M_6
1,2,3,4	1.872567	4.084438
1,2,3,5	1.758934	41.559028
1,2,3,6	1.500378	6.901214
1,2,4,5	2.045049	2.752602
1,2,4,6	1.787842	11.106032
1,2,5,6	1.664384	4.693668
1,3,4,5	2.190959	2.380921
1,3,4,6	1.987354	5.58962
1,3,5,6	1.893959	4.303475
1,4,5,6	2.147044	6.74884
2,3,4,5	2.151589	4.654878
2,3,4,6	1.923324	3.567641
2,3,5,6	1.772669	8.321244
2,4,5,6	2.074924	2.622965
3,4,5,6	2.225193	2.35483

The combination indicates which legs are chosen as reference legs 1, 2, 3, and 4 among the six possible legs.

51, 51, 40, and 46 s. If we have eliminated only the combination having the worst M_6 (1,2,3,5) the worst computation time will have been 329 s for combination (3,4,5,6) (which has the worst M_s), then 275 s for combination (1,3,4,5) (which has the second worst M_s) and finally 117 s for combination (2,4,5,6).

10. Example 3

In this section we consider the INRIA "left-hand" parallel robot that has been presented in numerous papers. Our algorithm has been tested for all 64 combinations of leg lengths obtained when each leg length is either the minimal or maximal possible joint limits 52.249, 55.749. For 14 combinations among the 64 combinations, there is no solution for the FK problem. On a single computer, the average computation time for solving the FK problem is about 20 s with a minimum of 2 s and a maximum of 50 s.

Note that this algorithm is very efficient. We have submitted this problem to `NUMERICA` (Van Hentenryck, Michel, and Deville 1997), a classical constraints-based solver, without getting any solution after hours of computation. We have also used a general solver implemented in `ALIAS` which was able to find the solutions in about 1 h.

11. Real-Time Operator

As mentioned previously, our algorithm is not intended to be used in a real-time context. However, in this case we may assume safely that the determination of the pose of the platform at time t_k may rely first on a similar calculation performed at time t_{k-1} having given a pose $P_{k-1} = \{B_1^{k-1}, \ldots, B_n^{k-1}\}$. Furthermore, we may assume that upper bounds V_{max}, Ω_{max} on the velocities and angular velocities of the robot are known. It is then easy to deduce from P_{k-1} a bounding box for P_k. For each coordinate of B_j^k, the box will be centered at B_j^{k-1} and its edge will have length $2(t_k - t_{k-1})(V_{max} + \Omega_{max}||B_jB_1||)$.

The bounding box will usually be much smaller than the bounding box used to determine all the solutions, and our experiment with the INRIA "left-hand" robot has shown that in that case the computation time is approximately similar to the usually used Newton scheme. Indeed, if the Newton scheme converges toward the current pose in most cases the unicity filter will determine that using the Newton scheme is safe. Hence, the only overhead will be due to this unicity test which amounts mostly to the numerical inversion of an $m \times m$ matrix. The computation time of this inversion is negligible as soon as the Newton scheme needs more than one iteration to converge.

However, at the same time our algorithm is safer:

- if only one solution is found, we guarantee that the solution is the current pose of the platform, while the Newton scheme may converge toward a solution of the forward kinematics that is not the current pose;
- if more than one solution is found, for example if the robot is close to a singularity, then it will be safer to immediately stop the robot as we cannot know for sure what is the current pose; the Newton scheme in that case may either fail to converge or provide a solution that is not the current pose, with severe consequences.

12. Conclusion

Interval analysis provides an interesting alternative for numerically solving the forward kinematics of parallel robots with the following advantages:

- certified result—all solutions will be provided so that they can be computed with a pre-selected accuracy and singular configurations are detected;
- the method can be adapted to take into account physical constraints with the advantage that the computation time will be reduced;
- it offers an alternative to real-time computation that is as competitive in terms of computation time but is safer than the Newton scheme;
- it allows us to take into account uncertainties in the model of the robot or in the measurements of the leg lengths.

Although the principle of interval analysis is quite straightforward we have shown that efficiency relies heavily on a set of filtering methods. Some of these methods are well known but they have been improved to take into account the structure of the FK equations and such a combination of methods has not been used in the past. The distance equation solver that has been presented here has also been used for the determination of conformation of molecules where the distance between the atoms are known (a molecule with 100 atoms has been successfully identified).

Acknowledgments

We are indebted to Pr. Neumaier for numerous discussions about systems solving and interval analysis that have allowed us to greatly improve parts of our algorithm. We would also like to thank the anonymous reviewers for their very useful comments.

References

Bouvet, D. and Garcia, G. 2001. Guaranteed 3-d mobile robot localization using an odometer, an automatic theodolite and indistinguishable landmarks. *IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 23–25, pp. 3612–3617.

Carreras, C. et al. 1999. Robot reliability estimation using interval methods. *Workshop on Applications of Interval Analysis to Systems and Control*, February, pp. 371–385.

Castellet, A. 1998. *Solving inverse kinematics problems using an interval method*. Ph.D. Thesis, Universitat Politechnica de Catalunya, Barcelona, Spain.

Chablat, D., Wenger, J., and Merlet, J.-P. 2002. Workspace analysis of the Orthoglide using interval analysis. *Proceedings of ARK*, Calle de Malavada, June 29–July 2, pp. 397–406.

Collavizza, M., Deloble, F., and Rueher, M. 1999. Comparing partial consistencies. *Reliable Computing* 5:1–16.

Didrit, O., Petitot, M., and Walter, E. 1998. Guaranteed solution of direct kinematic problems for general configurations of parallel manipulator. *IEEE Transactions on Robotics and Automation* 14(2):259–266.

Dietmaier, P. 1998. The Stewart–Gough platform of general geometry can have 40 real postures. *Proceedings of ARK*, Strobl, Austria, June 29–July 4, pp. 7–16.

Faugère, J. C. and Lazard, D. 1995. The combinatorial classes of parallel manipulators. *Mechanism and Machine Theory* 30(6):765–776.

Geist, A. et al. 1994. *PVM: Parallel Virtual Machine*, MIT Press, Cambridge, MA.

Hansen, E. 1992. *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.

Husty, M. L. 1996. An algorithm for solving the direct kinematic of Stewart–Gough-type platforms. *Mechanism and Machine Theory* 31(4):365–380.

Innocenti, C. 2001. Forward kinematics in polynomial form of the general Stewart platform. *ASME Journal of Mechanical Design* 123:254–260.

Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. 2001. *Applied Interval Analysis*, Springer-Verlag, Berlin.

Kiyoharu, T., Ohara, F., and Hiromasa, H. 2001. Fast interval bisection method for finding all solutions of nonlinear equations and its application to inverse kinematics for general manipulators. *Transactions of the Institute of Electrical Engineers of Japan, Part C* 120-C(4):590–596.

Lazard, D. 1992. Stewart platform and Gröbner basis. *Proceedings of ARK*, Ferrare, Italy, September 7–9, pp. 136–142.

Lee, T.-Y. and Shim, J.-K. 2001a. Forward kinematics of the general 6-6 Stewart platform using algebraic elimination. *Mechanism and Machine Theory* 36:1073–1085.

Lee, T.-Y. and Shim, J.-K. 2001b. Elimination-based solution method for the forward kinematics of the general Stewart–Gough platform. *Proceedings of the 2nd Workshop on Computational Kinematics*, F. C. Park and C. C. Iurascu, editors, May 20–22, pp. 259–267.

Liu, A.-X. and Yang, T.-L. 1995. Configuration analysis of a class of parallel structures using improved continuation. *9th World Congress on the Theory of Machines and Mechanisms*, Milan, Italy, August 30–September 2, pp. 155–158.

Merlet, J.-P. 1999. Determination of 6d workspaces of Gough-type parallel manipulator and comparison between different geometries. *International Journal of Robotics Research* 18(9):902–916.

Merlet, J.-P. and Daney, D. 2001. A formal-numerical approach to determine the presence of singularity within the workspace of a parallel robot. *Proceedings of the 2nd Workshop on Computational Kinematics*, F. C. Park and C. C. Iurascu, editors, Seoul, South Korea, May 20–22, pp. 167–176.

Moore, R. E. 1979. *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics.

Neumaier, A. 1990. *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, UK.

Neumaier, A. 2001. *Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, UK.

Piazzi, A. and Visioli, A. 1997. A global optimization approach to trajectory planning for industrial robots. *International Conference on Intelligent Robotics and Systems (IROS)*, Grenoble, France, pp. 1553–1559.

Raghavan, M. 1991. The Stewart platform of general geometry has 40 Iconfigurations. *ASME Design and Automation Conference*, Chicago, IL, September 22–25, Vol. 32-2, pp. 397–402.

Ratscheck, H. and Rokne, J. 1995. Interval methods. *Handbook of Global Optimization*, R. Horst and P. M. Pardalos, editors, Kluwer, Dordrecht, pp. 751–819.

Revol, N. and Rouillier, F. 2002. Motivations for an arbitrary precision interval arithmetics and the MPFI library. *Validated Computing Conference*, Toronto, Canada, May 23–25.

Ronga, F. and Vust, T. 1992. Stewart platforms without computer? *Conference on Real Analytic and Algebraic Geometry*, Trento, Italy, pp. 97–212.

Rouillier, F. 1995. Real roots counting for some robotics problems. *Computational Kinematics*, J.-P. Merlet and B. Ravani, editors, Kluwer, Dordrecht, pp. 73–82.

Rouillier, F. 2003. Efficient real solutions and robotics. *First EMS–SMAI–SMF Joint Conference on Applied Mathematics and Applications of Mathematics*, Nice, France, February 10–13.

Sreenivasan, S. V. and Nanua, P. 1992. Solution of the direct position kinematics problem of the general Stewart platform using advanced polynomial continuation. *22nd Biennial Mechanisms Conference*, Scottsdale, AZ, September 13–16, pp. 99–106.

Tagawa, K., Ohara, F., Ohta, Y., and Haneda, H. 1999. Direct inverse kinematics using fast interval bisection method and its automatic programming. *ICAR*, Tokyo, Japan, pp. 497–502.

Tagawa, K. et al. 2001. Optimal configuration problem of redundant arms considering endpoint compliance and its solution using interval analysis. *Transactions of the Society of Instrument and Control Engineers* 37(10).

Tapia, R. A. 1971. The Kantorovitch theorem for Newton's method. *American Mathematic Monthly* 78(1.ea):389–392.

Van Hentenryck, P., Michel, L., and Deville, Y. 1997. *Numerica: A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA.

Wampler, C. W. 1996. Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates. *Mechanism and Machine Theory* 31(3):331–337.

Yamamura, K., Kawata, H., and Tokue, A. 1998. Interval solution of nonlinear equations using linear programming. *BIT* 38(1):186–199.