AN ALGORITHM FOR THE SOLUTION OF INVERSE KINEMATICS PROBLEMS BASED ON AN INTERVAL METHOD

A. CASTELLET AND F. THOMAS
Institut de Robòtica i Informàtica Industrial,
Edifici Nexus, Gran Capità, 2–4, planta 2a.,
08034 Barcelona, Spain,
email: acastellet@iri.upc.es & fthomas@iri.upc.es

Abstract. In this paper we describe a general procedure to solve the positional inverse kinematics problem using an interval method. The classical interval Newton method is used together with specific developed interval cuts over the closure equations. The algorithm uses a basic branch-and-bound procedure to obtain all solutions of the inverse positional problem of arbitrary single-loop kinematic chains. Some preliminary examples are given, although still much improvement can be done in the heuristics involved and the code optimization, leaving plenty of room for future improvements.

1. Introduction

Solving loops of kinematic constraints is a basic requirement when dealing with inverse kinematics, task-level robot programming, assembly planning, or constraint-based modeling problems. This problem is difficult due to its inherent computational complexity (i.e., it is NP-complete) and due to the numerical issues involved to guarantee correctness and to ensure termination.

Two basic approaches have been used for solving this problem: continuation and elimination methods (Roth, 1994). Recently, interval methods for solving systems of non-linear equations have attracted much attention and have been explored by a variety of authors (Hansen, 1992) and (Kearfott, 1996). They have already been used to solve some kinematic problems proving to be robust but sometimes slow compared to continuation methods (Van Hentenryck, McAllester and Kapur, 1997).

In our case, the interval method receives a box, i.e. an interval-tuple of the variables of rotation and translation, specifying the initial bounds. Then, it returns a set of boxes containing the different solutions. When the kinematic chain is redundant, this method is also able to provide a discretized version of the underlying self-motion manifold.

This paper is structured as follows: Section 2 describes the closure equations used, Section 3 gives an introduction to interval methods, in Section 4, the basic steps of the algorithm are described and, in Section 5, three different examples are presented.

2. Closure Equations

The n-bar mechanism used here was first introduced in (Thomas, 1992) and is defined as a closed single-loop mechanism composed of n links –or bars–, each one being orthogonal to the next bar and having a rotational and a translational degree of freedom (Fig.1).

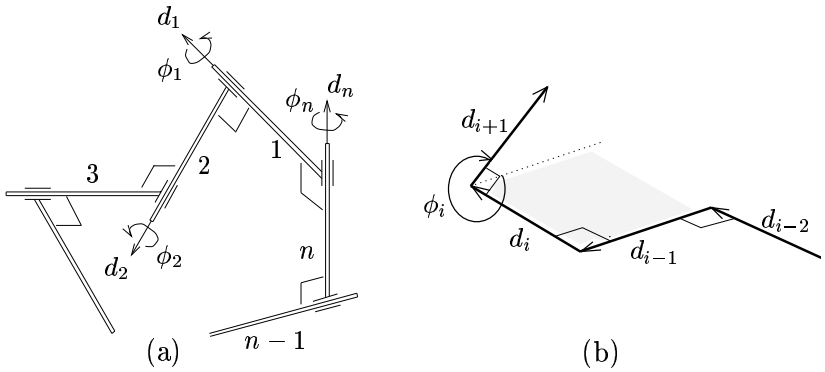

Figure 1. The n-bar mechanism (a) and the involved degrees of freedom (b).

The loop equation of the n-bar mechanism can be expressed as

$$\prod_{i=1}^{n} \mathbf{T}(d_i)\mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I}\,, \tag{1}$$

where $\mathbf{T}(d_i)$ is a translation along the x-axis, $\mathbf{R}(\phi_i)$ a rotation around the x-axis and $\mathbf{Z}$ a rotation of $\pi/2$ around the z-axis. The n-bar mechanism is defined by the *vector of rotations*, $\boldsymbol{\phi} \triangleq (\phi_1, \phi_2, \ldots, \phi_n)$, and the *vector of translations*, $\mathbf{d} \triangleq (d_1, d_2, \ldots, d_n)$.

Any single-loop kinematic chain can be described using an n-bar mechanism. We have been using it as an alternative to the Denavit-Hartenberg

parameters to represent spatial mechanisms because of its simple structure, which leads to more regular closure equations than using the D-H parameters.

A deep study of the n-bar mechanism and its underlying self-motion manifold can be found in (Castellet and Thomas, 1998). Here, we only give two equivalent sets of closure equations, which are used by our interval method.

The loop equation (1) can be factored into the following two equations (Thomas, 1992):

$$\mathbf{F}(\boldsymbol{\phi}) \stackrel{\triangle}{=} \mathbf{A}_1^n(\boldsymbol{\phi}) = \mathbf{I} \tag{2}$$

and

$$\mathbf{T}(\boldsymbol{\phi}, \mathbf{d}) \stackrel{\triangle}{=} \sum_{i=1}^{n} \left[\mathbf{A}_1^{i-1}(\boldsymbol{\phi}) \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} \right] = \mathbf{0} \,, \tag{3}$$

where $$\mathbf{A}_k^l(\boldsymbol{\phi}) \stackrel{\triangle}{=} \begin{cases} \mathbf{I} & \text{for } k = l+1 \\ \prod_{j=k}^{l} \mathbf{R}(\phi_j)\mathbf{Z} & \text{for } k \leq l \end{cases} .$$

These two equations are called the rotation and the translation equations, respectively. It is straightforward to proof that the solutions to both of them are the solutions to the loop equation (1).

Parameterizing the solution sets of both matrix equations, we can derive another set of closure equations, the *parametric rotation and translation equations*:

$$\begin{aligned} \phi_{n-2} &= r_1(\phi_1, \phi_2, \ldots, \phi_{n-3})(+\pi) \\ \phi_{n-1} &= \pm r_2(\phi_1, \phi_2, \ldots, \phi_{n-3}) \\ \phi_n &= r_3(\phi_1, \phi_2, \ldots, \phi_{n-3})(+\pi) \\ \\ d_{n-2} &= t_1(\phi_1, \phi_2, \ldots, \phi_n, d_1, d_2, \ldots, d_{n-3}) \\ d_{n-1} &= t_2(\phi_1, \phi_2, \ldots, \phi_n, d_1, d_2, \ldots, d_{n-3}) \\ d_n &= t_3(\phi_1, \phi_2, \ldots, \phi_n, d_1, d_2, \ldots, d_{n-3}) \,. \end{aligned} \tag{4}$$

The expressions of r_i and t_i are explicitly given in (Castellet and Thomas, 1998). They are simple products and sums of rotation matrices affected by an atan2 function for the r_i. There are two possible solutions for the rotation equations, corresponding to the π added to r_1 and r_3 and the $\pm$ sign in r_2. Note that these equations are general for any kinematic chain represented by an n-bar mechanism.

These equations are also equivalent to the loop equation (1), except for points corresponding to singularities of the parameterization; that is, when $\phi_{n-1} = 0$ or π. Expressions of their derivatives are also simple and can be found in (Castellet and Thomas, 1998).

3. Interval Methods

Interval methods manipulate upper and lower bounds on variables and are based on interval arithmetic (Hansen, 1992). They have been used to solve systems of nonlinear equations, global optimization problems and to avoid rounding errors due to floating-point representations of real numbers in computers.

One of the problems that arise when using interval arithmetic is the *overestimation* of functions where a variable appears more than once. This overestimation is due to the fact that a natural evaluation of the function supposes that the variable varies independently in each appearance.

For our purposes we need to solve the system of nonlinear equations derived from the closure equations. We describe it as

$$F(\mathbf{x}) = 0 \ , \ \text{where} \quad F : \mathbf{x} \subset \mathbb{R}^n \to \mathbb{R}^m \ .$$

An interval-method algorithm would receive a box, i.e. two interval vectors specifying the initial range of the vectors of rotation and translation, and it would return a set of boxes of specified accuracy containing all solutions.

Interval Newton Methods have been widely used to solve systems of nonlinear equations and we describe them briefly in next section (for more details see (Kearfott, 1996)). Here, an interval Newton method is used together with some specific developed interval cuts, as described in Section 3.2. Then, in Section 3.3, we describe the branch-and-bound strategy required for the algorithm to converge.

3.1. INTERVAL NEWTON METHODS

Interval Newton methods can be viewed as computational analogues of the Brouwer fixed-point theorem. The first-order Taylor expansion around the center of the box $\check{\mathbf{x}}$ results in a interval linear system that must be bounded:

$$\mathbf{S}(\mathbf{x} - \check{\mathbf{x}}) = -F(\check{\mathbf{x}}) \ ,$$

where $\mathbf{S}$ is a sort of interval jacobian matrix of the original system. The computation of this matrix is an important point in interval Newton methods. Its efficiency can be highly improved using slope functions instead of interval derivatives. Although both slope functions and interval derivatives can be obtained by automatic differentiation, we use directly the explicit

expressions we have for the derivatives of the closure equations to compute a Hansen's slope matrix (Hansen, 1992).

The most suitable method to compute outer estimates to the solution set of an interval linear system is the *interval Gauss-Seidel method*. It proceeds coordinate by coordinate in a similar way than its real counterpart, but usually requires the system to be preconditioned in order to be effective.

Preconditioning the system is one of the clues of interval Newton methods. Although the mostly used preconditioning matrix is the midpoint inverse of the system, often, much better preconditioners can be found. Families of *optimal preconditioners* have been developed, which optimize some specific criterion. The most studied, and the one we use, is the width-optimal LP-preconditioner, which minimizes the width of the resulting intervals. Using some heuristics, this preconditioner can be computed as a linear-programming problem.

Finally, existence and uniqueness in the interval Gauss-Seidel method can be usually verified without extra computing, since they result as a byproduct of the algorithm.

3.2. SPECIFIC CUTS

Interval cuts are procedures which operate on a set of constraints and a box, reducing this box by deriving a new bound on one of the variables. In (Van Hentenryck, McAllester and Kapur, 1997), three general cuts are described, which operate in a similar way than the interval Newton method.

We have developed some specific cuts, based on the structure of the rotation and the translation equations (2) and (3). In both of these equations, it is possible to isolate the variable we want to cut and evaluate directly its range of possible values.

We can perform three different types of specific cuts: two cuts for the variables of rotation –derived from both the rotation and the translation equation– and one cut for the variables of translation –derived from the translation equation. All these cuts are described in (Castellet and Thomas, 1997); we include here only their final expressions:

1. Cutting ϕ_i with the rotation equation:

$$\phi_i \leftarrow \arccos(v^i_{22} \cap v^i_{33}) \cap \arcsin(v^i_{32} \cap -v^i_{23}) \cap \phi_i \ , \tag{5}$$

   where $\mathbf{V}^i \triangleq (\mathbf{Z}\mathbf{A}^n_{i+1}(\boldsymbol{\phi})\mathbf{A}^{i-1}_1(\boldsymbol{\phi}))^t$.
2. Cutting ϕ_i with the translation equation:

$$\phi_i \leftarrow \arcsin\left(\frac{w^i_{03}w^i_{11} - w^i_{13}w^i_{02}}{{w^i_{11}}^2 + {w^i_{13}}^2}\right) \bigcap \arccos\left(\frac{w^i_{02}w^i_{11} - w^i_{13}w^i_{03}}{{w^i_{11}}^2 + {w^i_{13}}^2}\right) \bigcap \phi_i \ , \tag{6}$$

where $\mathbf{w}_0^i \stackrel{\triangle}{=} -\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t \sum_{k=1}^{i-1} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix}$ and

$\mathbf{w}_1^i \stackrel{\triangle}{=} \sum_{k=i+1}^{n} \mathbf{A}_{i+1}^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix}$.

3. Cutting d_i with the translation equation:

$$d_i \leftarrow \left(\mathbf{w}_0^i - \mathbf{A}_i^i(\boldsymbol{\phi})\mathbf{w}_1^i\right) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \cap d_i \ . \tag{7}$$

These specific cuts are computationally much less expensive than performing an interval Newton cut. They are usually able to cut large initial boxes at first stages, but they are not enough –by themselves– to converge to the solutions. Thus, they should be seen as complementary to the interval Newton method and the branch-and-bound strategy described in the next section.

3.3. BRANCH-AND-BOUND STRATEGY

After reducing a box using our specific cuts and the interval Newton method, three possibilities arise. First, the pruning operation may have resulted in an empty box, in which case there are no solutions. Second, it may be the case that the interval associated with each variable has reached a width below a specified accuracy. In this case we terminate and return the box containing a solution, depending on the existence test. If the pruning operation results in a box which is not of sufficient accuracy, we split the box and two branches are generated. Then, solutions on each branch are recursively searched.

Replacing a single box by two smaller boxes allows either the overestimation in interval extensions to be reduced or the interval Newton method to converge in the smaller boxes.

Generalized bisection divides a box by one of its variables. A natural choice for the bisection coordinate would be the variable for which the width is maximum. However, the variable corresponding to maximum width is not always the most effective at reducing overestimation or producing boxes in which the interval Newton method will converge. Many heuristics have been proposed to chose the variable to split (Kearfott, 1996); they usually attempt to choose a coordinate such that the function components have the highest variations along the range of the variable.

4. General Algorithm

The overall algorithm combines an interval Newton method with our specific cuts and the branch-and-bound strategy in order to isolate all solutions into boxes as small as desired.

The input of the algorithm is an initial box $\mathbf{x}_0$, the desired accuracy ϵ and the maximum number of boxes to process M. The box $\mathbf{x}_0$ is composed of the vectors of rotations and translations. Most of the elements of these vectors are degenerate intervals of zero width corresponding to the geometry of the mechanism. The other elements are those representing the configuration of the mechanism (rotations and translations of joints) and are intervals whose limits are the allowed range of joint motions.

The algorithm can be summarized in the following steps:

- place the initial box $\mathbf{x}_0$ onto an empty list $\mathcal{L}$
- DO $k = 1$ to M WHILE $\mathcal{L} \neq \emptyset$
    1. remove one box from $\mathcal{L}$ and place it in the current box $\mathbf{x}_c$
    2. DO WHILE size of $\mathbf{x}_c > \epsilon$
        a. *(reduce the box using specific cuts)*
            FOR $i = 1$ TO number of variables DO (specific cuts)
                IF variable i is a rotation THEN
                    cut variable i using cut (5)
                    cut variable i using all possible cuts (6)
                ELSE (variable i is a translation)
                    cut variable i using all possible cuts (7)
        b. *(reduce the box using an interval Newton method)*
            DO $j = 1$ to 10 WHILE size of $\mathbf{x}_c > \epsilon$
                FOR $i = 1$ TO number of variables DO
                    compute the optimal preconditioner row
                    perform a Gauss-Seidel iteration for variable i using the closure equations (4)
            END DO
        c. IF the box does not contain a solution THEN
                EXIT loop 2 and CYCLE main loop
        d. IF $\mathbf{x}_c$ has not been reduced THEN
                bisect $\mathbf{x}_c$ and insert one of the boxes into $\mathcal{L}$, while the other one becomes $\mathbf{x}_c$
        END DO
    3. insert $\mathbf{x}_c$ into list $\mathcal{R}$ (list of resulting boxes)
- END DO

Although the algorithm converges to the solutions, many heuristics can be introduced which may speed up the process considerably.

Step 2.a. could be iterated until specific cuts do no longer reduce the box. However, experiments show that the reduction after the first iteration decreases significantly and it is not worth cycling more.

It should be pointed out that in Step 2.b., the interval Newton method has to be applyed to both sets of equations (4). Here, more sweeps are useful and we limit them to 10 to avoid cases with slow convergence.

5. Examples

The experiments we have been doing show that the main problem is that the algorithm usually converges to a solution without branching only if the box is *very* small. We have been trying to avoid branching as much as possible in order to speed up the process. Using Hansen's slopes and width-optimal linear programming preconditioners in the interval Newton method is crucial, but other minor heuristics have been used, such as the ones in the computation of the preconditioner (Castellet, 1998).

The algorithm has been implemented in C++ using BIAS/PROFIL portable interval libraries (Knüppel, 1993) in a SUN Ultra 2 2300 Creator with a 296MHz processor.

We show here three examples: a PUMA 560 in three different configurations, a particular 6R mechanism also in three different configurations and a case of the 7R example of Duffy (1980) and Lee (1988). Although there are no aparent differences with the other examples, in the last case the program takes a long time to obtain the right solutions.

For each of these examples, we give the number of branchings, the number of boxes eliminated by specific cuts and by an interval Newton iteration, the average sweeps of the interval Newton methods and the required CPU time. All examples return 8 solutions, except for Duffy's example, which returns 6.

PUMA 560. We use 14 bars to describe a PUMA 560 and 3 more bars to close the chain. The vectors of rotations and translations of a PUMA 560 are:

$$\boldsymbol{\phi} = (90, \theta_1, -90, \theta_2, 180, \theta_3, 90, \theta_4, 90, \theta_5, 90, \theta_6, 90, 0, \phi_{15}, \phi_{16}, \phi_{17})$$
$$\mathbf{d} = (0, 0, 0, 0, 432, 149.098, -20.5, 433, 0, 0, 0, 56.5, 0, 0, d_{15}, d_{16}, d_{17})$$

A 6R. We use 12 bars to describe a 6R mechanism and 3 more bars to close the chain. The vectors of rotations and translations are:

$$\boldsymbol{\phi} = (90, \theta_1, 90, \theta_2, 90, \theta_3, 90, \theta_4, 90, \theta_5, 90, \theta_6, \phi_{13}, \phi_{14}, \phi_{15})$$
$$\mathbf{d} = (5, 2, 7, 4, 8, 0, 2, 0, 12, 15, 6, 3, d_{15}, d_{16}, d_{17})$$

TABLE 1. PUMA 560

Configuration	Branching	specific cuts	Newton	avg. New. sweeps	CPU time
$\boldsymbol{\phi} = (\ldots, 195, 90, 335)$ $\mathbf{d} = (\ldots, -780, -15, 430)$	236	159	71	1.88	1'24"
$\boldsymbol{\phi} = (\ldots, 200, 56, 305)$ $\mathbf{d} = (\ldots, -370, 600, -550)$	197	154	37	2.40	1'15"
$\boldsymbol{\phi} = (\ldots, 178, 52, 279)$ $\mathbf{d} = (\ldots, 340, -200, 615)$	328	282	40	1.78	1'42"

TABLE 2. A 6R

Configuration	Branching	specific cuts	Newton	avg. New. sweeps	CPU time
$\boldsymbol{\phi} = (\ldots, 200, 15, 85)$ $\mathbf{d} = (\ldots, 9, 0, 7)$	1 636	1 484	152	1.68	6'15"
$\boldsymbol{\phi} = (\ldots, 300, 340, 5)$ $\mathbf{d} = (\ldots, 1, 7, 5)$	1 983	1 804	179	1.79	7'52"
$\boldsymbol{\phi} = (\ldots, 20, 30, 340)$ $\mathbf{d} = (\ldots, 6, 8, 10)$	1 862	1 755	107	1.24	6'25"

Duffy's example. We need here 14 bars. The vectors of rotations and translations are:

$$\boldsymbol{\phi} = (\theta_1, 260, \theta_2, 273, \theta_3, 300, \theta_4, 300, \theta_5, 273, \theta_6, 260, \theta_7, 215)$$
$$\mathbf{d} = (1.0, 0.9, 1.2, 1.1, 0.8, 1.5, 2.0, 1.5, 0.8, 1.1, 1.2, 0.9, 1.0, 0.5)$$

TABLE 3. Duffy's example

Configuration	Branching	specific cuts	Newton	avg. New. sweeps	CPU time
$\theta_7 = 30$	11 264	9 977	1 284	1.53	38'25"

6. Conclusions

In this paper we have summarized the basic ideas of a general algorithm based on an interval method for solving single-loop inverse kinematics problems. The procedure returns all solutions and works even in singular situations and for mechanisms with special geometries. Although the algorithm is still too slow for most practical applications, much improvement can be done in the heuristics involved and in the code optimization.

Because of its generality, interval methods can be seen as a competitive alternative to elimination and continuation methods for inverse kinematics problems, despite the slowness of these preliminary results.

The contents of this paper is fully developed in (Castellet, 1998).

7. Acknowledgements

The research work reported here has been partially supported by the Spanish CICYT under contract TIC96-0721-C02-01.

References

Castellet, A., (1998), Solving Inverse Spatial Kinematic Problems Using an Interval Method, PhD thesis, Universitat Politècnica de Catalunya, Spain, (to appear).

Castellet, A., and Thomas, F., (1997), Using Interval Methods for Solving Inverse Kinematic Problems, Proc. Computational Methods in Mechanisms, NATO ASI, Varna, Bulgaria **vol. 2**, pp. 135–144.

Castellet, A., and Thomas, F., (1998), Characterization of the Self-Motion Set of the Orthogonal Spherical Mechanism, Mech. Mach. Theory, (in print).

Duffy, J., and Crane, C., (1980), A Displacement Analysis of the General Spatial 7-Link, 7R Mechanism, Mech. Mach. Theory, **vol. 15**, pp. 153–169.

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

Kearfott, R. B., (1996), *Rigorous Global Search: Continuous Problems*, Kluwer Ac. Pub.

Knüppel, O., (1993), PROFIL-Programmer's Runtime Optimized Fast Interval Libraries, Tech. Report, Technische Universität Hamburg-Harburg, 93.4., available at http://www.ti3.tu-harburg.de/indexEnglisch.html.

Lee, H.-Y., and Liang, C.-G., (1988), Displacement Analysis of the General Spatial 7-Link 7R Mechanism, Mech. Mach. Theory, **vol. 23**, n. 3, pp. 219–226.

Roth, B., (1994), Computational Advances in Robot Kinematics, *Advances in Robot Kinematics and Computational Geometry*, eds. A. J. Lenarčič and B. B. Ravani, Kluwer Ac. Pub., pp. 7-16.

Thomas, F., (1992), On the N-bar Mechanism, or How to Find Global Solutions to Redundant Single Loop Kinematic Chains, Proc. IEEE Int. Conf. Robotics Automat **vol. 1**, pp. 622–634.

Van Hentenryck, P., McAllester, D., and Kapur, D., (1997), Solving Polynomial Systems Using a Branch and Prune Approach, SIAM J. Numer. Anal., **vol. 34**, no. 2, pp. 797–827.